FROM THE EDITOR

EDITOR

Carlis Collins
editor@xilinx.com
408-879-4519

SENIOR DESIGNER

Jack Farage

BOARD OF ADVISORS

Dave Stieg
Dave Galli
Mike Seither
Peter Alfke

Pronto Creative
www.prontocreative.com

Xcell

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3450
Phone: 408-559-7778
FAX: 408-879-4780

A Tale of Two Giants...

Once upon a time there were two electronic giants, struggling for dominance in the Valley of Silicon. They were unfriendly rivals; each making similar products, each holding on to an equal share of the marketplace, each "leapfrogging" the other with new designs. When Giant X introduced a new MP3 player, for example, Giant Y would soon introduce one with more features at a lower cost. And, because they used the same basic components, design methods, and manufacturing processes, they each remained "competitive," but their profit margins were atrocious. The competition was fierce; life was uncertain; no one smiled.

Then one bright morning (after reading a particularly insightful editorial in the Xcell Journal), Giant X awoke from a vivid dream in which he saw the future. He thought "What if I could create new products that never had to be replaced? What if I could sell customized features, options, and complete new designs, and download them to my customers, anywhere in the world, over the Internet? That's like manufacturing something once, but selling it many times over!" He knew it was the next "big thing." He grinned a big toothy grin.

Giant X began designing all his new products using the latest programmable logic technology (from Xilinx of course). The new FPGAs and CPLDs were amazing; they were dense, fast, inexpensive, and required little power; they were easy to use because the development tools were fast and efficient, and there was a lot of Intellectual Property (cores) available to make his life easy. Plus, with Xilinx technology, he could provide an Internet interface to all of his products and easily download almost any new design the market demanded. "Simply brilliant!" remarked the press. "Amazing!" remarked his customers. "Highly profitable!" said his shareholders. Everyone grinned big toothy grins, except Giant Y of course.

It wasn't long before Giant Y's market share began to nosedive. In a panic he worked night and day to keep up with the almost daily introduction of new products, features, and options from Giant X; but he could no longer compete using his old technology. Greatly embarrassed, Giant Y quietly packed his bags and left town; he was never heard from again.

The moral of this story is clear. Low-cost, high-performance programmable logic, reprogrammed remotely, is the obvious next step in the evolution of logic design—the advantages are overwhelming.

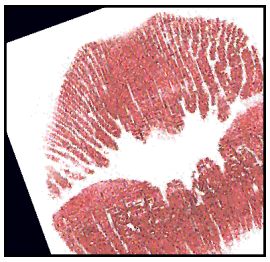
This issue of Xcell will show you some of our latest, low-cost, remotely-reprogrammable giant killers.

ARTICLES

COLUMN

4

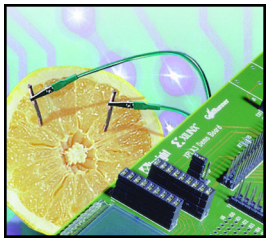
The year 2000 promises to bring even faster growth, with many new applications, because our technology has passed a key milestone.



COVER STORY

5

Spartan™ FPGAs are experiencing tremendous growth due to their inherent advantages over ASICs.



PRODUCT INFORMATION

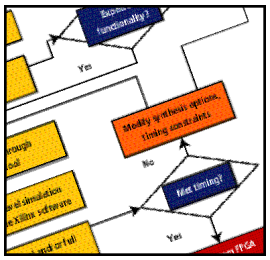
26

New CoolRunner devices are ideal for low-power, high-performance applications.

APPLICATIONS

35

CoolRunner™ CPLDs are used as the main controller for an MP3 player. (Also see a Spartan-II MP3 design on page 15.)



SOFTWARE

52

This HDL design methodology can help you use the largest Virtex™ FPGAs with a minimum amount of time spent on synthesis, simulation, and verification.

www.xilinx.com

Inside This Issue:

For A FREE Subscription To The Xcell Journal

E-mail your request to: literature@xilinx.com, please be sure to include:

1. Your full name and mailing address
2. Your title
3. The name of your company
4. Your e-mail address
5. Is this new subscription or a subscription renewal?

Xcell

The Programmable Logic Market Grew Significantly in 1999.

...and the year 2000 promises to bring even faster growth, with many new applications, because our technology has passed a key milestone.

by Wim Roelandts, President and CEO, Xilinx

Programmable logic technology has made many significant advances over the last year, and is well on its way to becoming the development technology of choice for most applications. Xilinx is growing very quickly as a result. Here are some of the highlights of our phenomenal growth in 1999:

- The Xilinx stock split twice, once in March and once in December. This helped give us a market capitalization of about $14.5 billion, making us second only to Intel among semiconductor suppliers here in Silicon Valley. This gives us an increased ability to fund new research and bring advanced new technologies to market.
- Xilinx was placed on the Merrill Lynch "Top 10 Tech" list (replacing Intel), indicating their highest confidence in our continued strength, and in the growth potential of the overall programmable logic market.
- Xilinx was added to the S&P 500 index, another indication of the market's confidence in our ongoing strength.
- We built a new 180,000 sq. ft. building on our San Jose campus to handle our recent growth, and began the development of a new 130,000 sq. ft. facility in Longmont Colorado. Groundbreaking is scheduled for March 2000.
- We expect our revenues to approach $1 billion by the end of our fiscal year (March 2000), 50% higher than last year.

It was a very good year for Xilinx; our investments in new device and software technologies have paid huge dividends, and our products have consistently extended the previous limits of density and performance while significantly reducing prices.

FPGAs were once used primarily in high-end, low-volume equipment which consists of about 25% of the total marketplace. This was due in part to the relatively high cost of FPGAs (as compared to custom devices). To serve the other 75% of the marketplace, our research showed us that we would need to produce high-performance, full-featured FPGAs with at least 100K gates, and sell those devices, in volume, for less than $10.00. With the introduction of our new Spartan-II family, that's exactly what we've done. This is a significant milestone that will bring many unique new applications.

The growth of the programmable logic marketplace is important to you because these devices allow you to get your products to market six to nine months faster than with ASICs, and they allow you to upgrade your equipment over any network after it's installed at your customers' locations. We here at Xilinx will continue our innovation and this will lead to higher densities, higher performance, more features, a wider selection of intellectual property (cores), and lower prices. We intend to keep making your job easier while adding value to your products.

THE NEW Spartan-II FPGA Family

KISS YOUR ASIC GOOD-BYE

Spartan FPGAs are experiencing tremendous growth due to their inherent advantages over ASICs.

by Jay Aggarwal, Product Marketing Manager
Spartan Series, Xilinx, jay.aggarwal@xilinx.com

The Spartan Series FPGAs, introduced by Xilinx in January 1998, offer a very attractive alternative to ASICs for your high volume, low cost applications. The Spartan families are designed to penetrate markets that were once dominated by ASICs. These include digital modems, printers, faxes, portable audio players (such as MP3), set-top boxes, and POS terminals. The Spartan Series is the first FPGA family to provide a complete and compelling mix of advanced features, low prices, high performance, and powerful development tools; the key ingredients required by ASIC designers. Now, the new Spartan-II™ FPGA family sets a new standard for low cost, and high performance.

The Spartan-II Family

Fabricated on a leading 0.18 µm, six-layer metal process, the Spartan-II family uses the most advanced process technologies available today. The family's core voltage operation is 2.5V, yet it incorporates unique I/O technology that allows both 3.3V and 5V I/O operation.

Device		XC2S15	XC2S30	XC2S50	XC2S100	XC2S150
System Gates		15K	30K	50K	100K	150K
Logic Cells		432	972	1728	2700	3888
Block RAMBits		16,384	24,576	32,768	40,960	49,152
Block RAMBlocks		4	6	8	10	12
DLLs		4	4	4	4	4
I/O Standards Supported		17	17	17	17	17
Max I/O		86	132	176	196	260
Packages	14x14mm	VQ100	VQ100			
	20x20mm	TQ144	TQ144	TQ144	TQ144	
	12x12mm	CS144	CS144			
	28x28mm		PQ208	PQ208	PQ208	PQ208
	17x17mm			FG256	FG256	FG256
	23x23mm				FG456	FG456

Table 1 - Integrated features.

The Spartan-II family includes new system-level features such as delay lock loops (DLLs), BlockRAM™, distributed RAM, multiple I/O standards, ultra-high performance, and power management. All of the features found in ASICs and ASSP devices are now available in the Spartan-II family at very attractive prices. Spartan-II FPGAs also provide an impressive array of highly complex cores (intellectual property) enabling you to further leverage the time-to-market benefits offered by programmable logic.

Memory

On-chip memory is vital to most designs, from buffering data between two dissimilar buses to

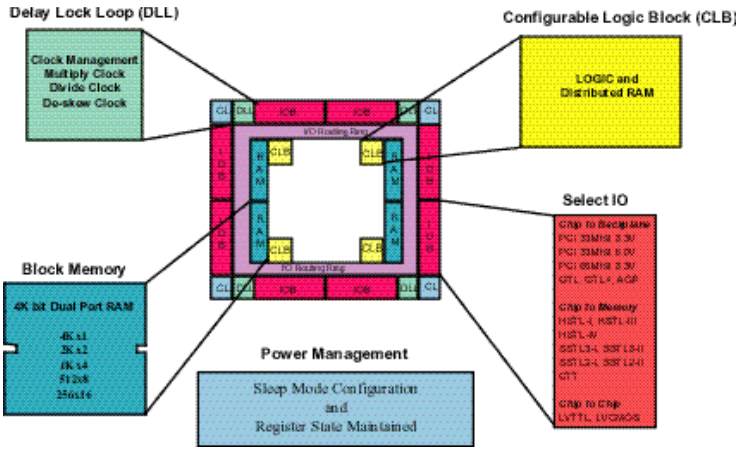


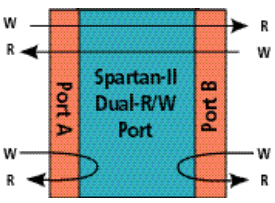
Figure 1 - Spartan-II family architecture.

providing storage locations of constants for high performance DSP functions. The Spartan-II family provides you with maximum memory flexibility.

Xilinx pioneered the capability of distributed memory (also found on Spartan and Spartan-XL families), which efficiently implements wide/shallow FIFOs or scratch pads memories. The family also incorporates BlockRAM (in blocks of 4Kbits) which efficiently implement memory for deep FIFOs, single port RAM, and True Dual Port RAMs as shown in Figure 2. Unlike competing two-port architectures, Xilinx provides true dual port RAM operation, for high-speed read and write operation.

Delay Locked Loops

DLLs perform the same tasks as traditional phase lock loops (PLLs) but are more robust and



Data Flow	Spartan II	Two-Port
A to B	Yes	Yes
B to A	Yes	No
A to A	Yes	No
B to B	Yes	No

Figure 2 - Spartan-II family dual port memory.

are less susceptible to noise interference, a common problem with PLLs. The Spartan-II DLLs allow you to multiply or divide the incoming clock on chip, as well as drive multiple clocks on the board. The DLL feature also allows you to de-skew the clock on chip, ensuring all nodes on the device are synchronized while providing minimum set-up and hold times.

With four DLLs per device, the Spartan-II family provides a sufficient number of DLLs with which you can perform multiple functions. For example, one DLL may be used to de-skew the clock on chip, one for multiplying the clock for accelerated performance on chip, two DLLs to drive clocks to various devices on the board—all at the same time. A single crystal oscillator and a Spartan-II device may provide all the clock management you need for your board design.

SelectI/O™

The Spartan-II family supports most of the popular and demanding I/O standards, including those that are optimized for high-speed memory interfaces. The new input/output standards that are supported, include SSTL, HSTL, AGP, GTL, GTL+, and PCI. The integration of these standards into the Spartan-II family now allow the elimination of costly bus transceivers that take up valuable board space.

The I/Os for the Spartan-II family are 5V and 3.3V tolerant for interfacing with older generation technology on the board. This is a significant advantage for the family because competing architectures are unable to support 5V operation.

Support for high-speed interfaces significantly increases performance over the Spartan (80 MHz) and Spartan-XL (100 MHz) families, to a blazing 200 MHz.

Power Management

Power consumption is an important issue, especially for portable and hand-held designs. The Spartan-II family extends power management features that were first established with the Spartan-XL family, which provides a power down pin on each device. Once activated, the device goes into a low power sleep mode in which power consumption is significantly reduced. When the pin is de-asserted, the device comes back into full power mode and retains its configuration as well as register states.

Pricing

The Spartan-II family has been created from the ground-up, in keeping with the Spartan Series philosophy, to provide industry-leading features, density, and performance at price points that match or beat ASICs and ASSP devices. The family has been able to achieve a milestone, long sought after and promised by the programmable logic industry but now only realized by the Spartan-II family: 100,000 gates for $10.00USD.

Below is a complete listing of high volume prices for the Spartan-II family at introduction.

Xilinx	Spartan II
Product	High Volume Price*
XC2S15	$ 3.95
XC2S30	$ 4.95
XC2S50	$ 7.95
XC2S100	$ 9.95
XC2S150	$ 12.95

* 250K units, a resale price, slowest speed/cheapest package

Table 2 - Spartan-II family pricing.

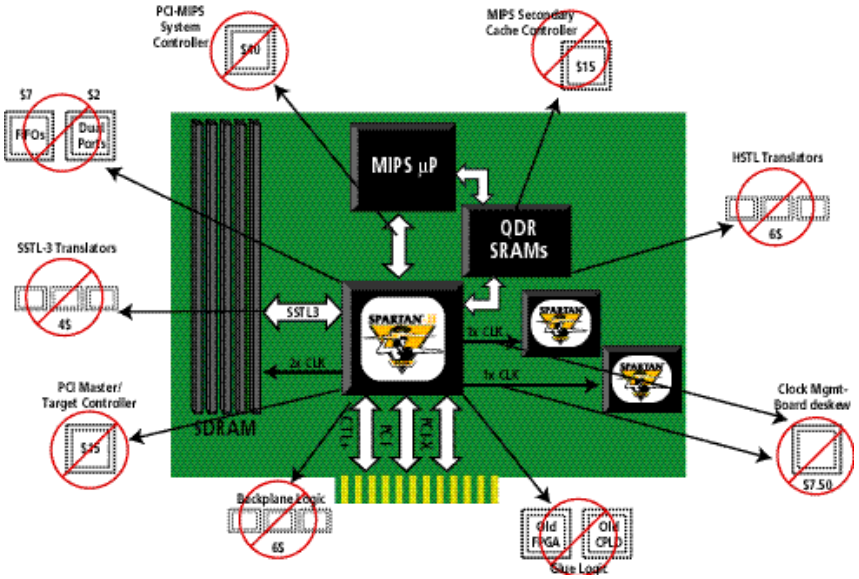


Figure 3 - Spartan-II value.

An Example of Spartan-II Value

The board diagram in Figure 3 illustrates how you can integrate many different functions into a Spartan-II FPGA to achieve significant cost savings. This example design includes a PCI master/target controller, some HSTL translators, a cache controller, SSTL-3 translators for SDRAM, a backplane interface, some glue logic, and the clock management device. All of these functions can be integrated into the XC2S100 Spartan-II device which costs just $10.00, almost two-thirds less than the discrete solution, with room to spare for more logic. The Spartan-II solution also uses less board real estate, requires less power, and provides higher reliability.

Conclusion

The Spartan-II family offers you the most cost-effective and flexible solution, enabling the fastest time-to-market with the lowest possible risk.

For more information regarding how the Spartan-II family addresses traditional ASIC and ASSP designs, please see the article on page 8.

New! Spartan-II FPGA Family

The Programmable ASSP

The Spartan-II family, combined with a vast portfolio of soft IP, is the first programmable logic solution to effectively penetrate the ASSP marketplace.

by Krishna Rangasayee, Manager, Strategic Applications,
Xilinx, krishna@xilinx.com

Spartan-II FPGAs offer more than 100,000 system gates at under $10.00 and are the most cost-effective PLD solution ever offered. They build on the capabilities of the very successful Virtex™ FPGA family and include all of the Virtex features, including SelectI/O™, BlockRAM™, Distributed RAM, and DLLs, with clock speeds up to 200 MHz.

PLDs Penetrating the ASSP Market

In the past, programmable logic devices had limited success in penetrating the ASSP market because they could not compete in the key areas of density, features, performance, and cost. However, the Spartan family competes very well due to the use of advanced process technologies. This approach has allowed Xilinx to significantly reduce die sizes, and therefore reduce the cost of the overall solution. This rapid process transition allows the Spartan family to compete with ASICs and ASSPs, and has opened up many new markets for PLDs.

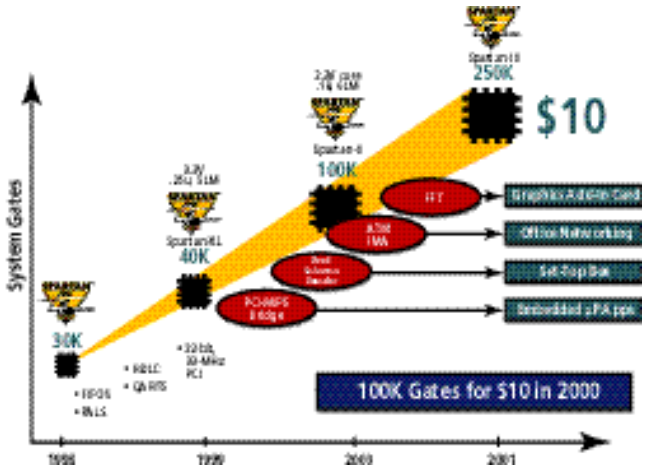


Figure 1 - PLD evolution - addressing the ASSP marketplace.

Advantages of a Programmable ASSP

A programmable ASSP like the Spartan-II family offers significant advantages over a stand-alone ASSP. The advantages are broadly classified under the following areas:

- The value of programmable ASSPs.
- Accommodating specification changes.
- Testing and verification.
- Xilinx Online™ - field upgradability.
- Problems in creating a stand-alone ASSP.

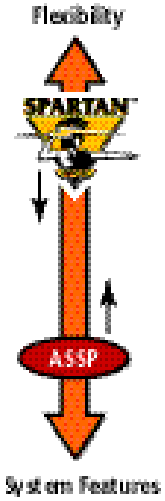
The Value of Programmable ASSPs

ASSPs, designed for a wide array of applications, are rarely able to meet your exact needs. With a programmable ASSP solution, such as Spartan-II FPGAs, you can choose the optimum feature set and optimize your design to achieve best possible results—this gives you a better design and saves money.

The PCI case study shown in Figure 3 is a good example. This Spartan-XL PCI solution was able to effectively cut the total product cost in half and also allow room to accommodate the extra logic that you may want to add to the backend PCI interface, such as a DMA controller, SDRAM controller, or FIFO.

Accommodating Specification Changes

ASSP vendors are motivated to quickly create solutions for emerging markets because of the high profit



- Spartan FPGAs migrate to higher densities to handle system features
  - –Maintaining low cost
- ASSPs attempt to offer flexibiity
  - –Differentiation need due to market pressures
  - –Available ASSPs require programmable logic
  - –Changing system standards
- ASSPs have large role in consumer, networking & data-processing
  - –Where Spartan FPGAs are successful
- PCI is the first successful ASSP competition

Figure 2 - Spartan-II family penetrates ASSP markets.

margins they stand to gain. However, the standards change constantly in these markets, often making ASSPs a risky choice. These conditions create many opportunities for the Spartan-II family, because with a Spartan device, you can upgrade your design to accommodate evolving specifications even after your systems are deployed in the field.

Testing and Verification

Another problem users encounter with stand-alone ASSPs is that the devices do not always behave as expected. Identifying problems is a lot easier with pro-

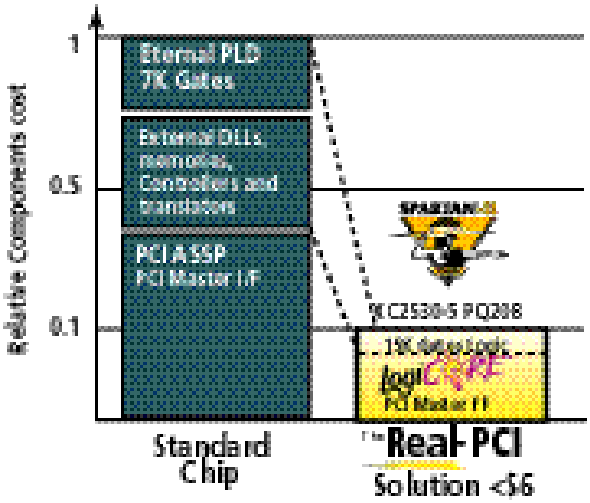


Figure 3 - Xilinx PCI solution vs. stand-alone ASSP.

grammable ASSPs, such as the Spartan-II FPGAs, because they are built on the fabric of a proven FPGA technology and the silicon has been pre-verified and guaranteed to perform. Because a programmable ASSP is inherently re-programmable, fixing any problem is simple. This is a tremendous value-added feature that a stand-alone ASSP cannot offer.

Xilinx Online for Field Upgradability

The Xilinx Online capability allows you to add new hardware features and fix bugs, over a network, without sending a technician to the field; this can add up to considerable maintenance and support savings over the entire life of the system. The value of field upgradability is illustrated in Figure 4.

Problems in Creating a Stand-alone ASSP

Vendors who create stand-alone ASSP devices must over-design their products to meet the requirements of a wide range of customers. A list of the various hurdles that an ASSP vendor faces today are:

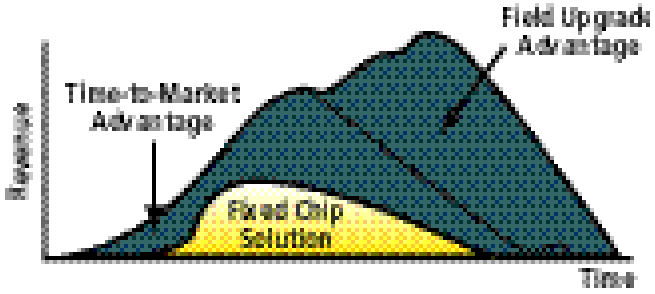


Figure 4 - Field upgradability extends the value of programmable ASSPs.

- **Choosing the Right ASSP** - The ASSP vendor must choose the right market segment.
- **Product Customization** - ASSP vendors face the challenge of creating one solution that must successfully meet the demands of a wide range of customers.
- **Development Cost and Amortization** - Stand-alone ASSPs have high NRE and engineering costs. These costs are increasing with process technology migration.

The Spartan-II family is unaffected by these hurdles and offers a cost-effective programmable ASSP solution.

Spartan-II ASSP Replacement Value

The Spartan-II family replaces and or competes against three classes of ASSPs, broadly classified as:

- Feature-Replacement ASSPs
- Logic-Replacement ASSPs
- Value-Added ASSPs

Feature-replacement ASSPs

Examples of "Feature-replacement ASSPs" are shown in Table 1. All of these functions are available in a Spartan-II FPGA without using any of the PLD's logic resources. Plus, the price of some of the Spartan-II devices is about the same as that of the ASSP they replace.

Feature Replacement ASSPs	Price
32-bit SSTL-3 Transceivers with Tristate Outputs	$ 4.00
32-bit to 64-bit HSTL-to-LVTTL Memory Address Latch	$ 6.00
32-bit LVTTL to GTL/GTL+ Transceivers with Live Insertion	$ 6.00
High Speed CMOS Digital PLLs	$ 1.00
High Speed Programmable Board Skew Clock Buffer	$ 7.50
2K x 8 Dual-Port Static RAM	$ 2.00
64,256,512,1K,2K,4K x 18 Synchronous FIFOs	$ 7.00
Hot Swap Controller	$ 2.00
Note:Pricing shown is approximate and for volumes of 100,000 units	

Table 1 - A list of potential feature replacement ASSPs replaced by the Spartan-II Family.

Logic-replacement ASSPs

Logic-replacement ASSPs are those that can be replaced by using the logic resources of a Spartan-II chip in combination with various IP cores. Examples of potential logic-replacement ASSPs are shown in Table 2.

Value-added ASSPs

Value-Added ASSPs fall into either of two categories:

- ASSPs that take unique advantage of the Xilinx architecture, like the ATM IMA devices from Applied Telecom. The class of field-upgradable ASSPs and network processors also fall into this category.
- ASSPs that serve emerging markets and markets that do not exist today, such as a PCI-X Master/Target Controller.

The Spartan-II family services all three classifications of ASSPs very well. Examples of Value Added ASSPs are shown in Table 3.

Logic Replacement ASSPs	Price
64-bit,66-MHz PCI v2.2 Bus Master	$ 25.00
32-bit,33-MHz CompactPCI(r) Bus Master Hot Swap Friendly PCI interface chip	$ 15.00
32-bit,33-MHz Bus Target chip	$ 12.00
32-bit,33-MHz PCI Master/Slave Controller	$ 14.00
32-bit,33-MHz PCI Target Controller	$ 12.00
STS-12C/STS-3C POS/ATM SONET Mapper	$ 120.00
PCI System Controller for 64-bit MIPS CPUs w/ Integrated SDRAM controller	$ 12.00
Advanced PCI System Controller for 64-bit MIPS CPUs	$ 40.00
Secondary Cache Controller for the R4600/R4700	$ 15.00
Low-Cost 8-Port 10/100 Fast Ethernet Switch Controller	$ 28.00
High Speed Microcontrollers are direct performance upgrades for the 8051	$ 8.00
256-Channel HDLC Controller	$ 60.00
Multi-Channel HDLC Controller with 32-bit, 66-MHz PCI Controller	$ 120.00
Block Floating Point 16 x 16 Complex Floating Point Multiplier	$ 300.00
Programmable FIR Filter	$ 310.00
Standalone FFT Processor	$ 450.00
Integrated Digital Switch	$ 12.00
HDLC Protocol Controller	$ 4.50
Multi Channel ATM AAL1 SAR	$ 90.00
Dual ADPCM Transcoder	$ 4.00
Integrated PCM Filter CODEC	$ 4.00
Viterbi with Reed-Solomon Decoder	$ 25.00
Reed-Solomon Forward Error Correction	$ 20.00
ALDC Data Compression	$ 12.00
DCLZ Compression	$ 22.00
ISDN Terminal Adapter with HDLC Controller	$ 10.00
Multichannel Network Interface Controller for HDLC	$ 60.00
Fast Ethernet (100 Mbps) Media Access Controllers (MAC)	$ 20.00
Note:Pricing shown is approximate and for volumes of 100,000 units	

Table 2 - A List of potential logic-replacement ASSPs supported by the Spartan-II family.

Value Added ASSPs	Price
64-bit,66-MHz PCI-X System Controller	NA
Quad ATM IMA Chip	$ 30.00
Octal ATM IMA Chip	$ 50.00
ARC Processor	NA
Note:Pricing shown is approximate and for volumes of 100,000 units	

Table 3 - A list of potential value added ASSPs supported by the Spartan-II Family.

Conclusion

The new Spartan-II FPGA family, due to its advanced features and low cost, is uniquely capable of replacing many standard ASSP devices. And though it may not replace all ASSPs, the Spartan family is now being used in many new high-volume, low-cost applications that were once dominated by stand-alone ASSPs.

The Spartan-II Design Flow
Simple, Powerful, Efficient

A design flow that offers distinct advantages when compared to an ASIC design methodology

by Craig N. Willert, Software Marketing Manager, Xilinx,
cnw@xilinx.com

With the rapid adoption of deep-submicron process technology in the design of FPGAs, Xilinx is now able to provide you with a cost-effective alternative to using an ASIC. But the cost of silicon is only one reason why the use of Spartan devices in high-volume applications is skyrocketing.

Spartan devices are designed using our robust suite of design tools. These tools have become rich in features as a result of the inclusion of a series of patented innovations. This article describes some of the key technologies that are included in the Xilinx development systems, including HDL optimized flows, timing driven layout, core (intellectual property) design and integration, and a comprehensive suite of verification tools.

Xilinx design methodology also offers some distinct advantages when compared to an ASIC design flow. These improvements in the art of logic design deliver benefits that reduce the cost of design development and accelerate time to market.

A Robust Suite of Design Tools

The development systems that Xilinx delivers today incorporate the collective innovations of over 15 years of programmable logic design expertise. These improvements to the programmable logic design process have resulted in the creation of a comprehensive, high quality suite of design tools. Xilinx development systems are packaged to deliver the best value for your dollar, enabling you to invest in just what you need to get the job done.

All Xilinx development systems include a comprehensive set of key technologies that enable the efficient design of powerful, high performance products. Other tools useful in the design of programmable logic are delivered as options or as part of the Foundation Series "packaged" solutions, including HDL design, simulation, synthesis, and optimization.

One of the benefits of programmable logic is the ability to program (and reprogram) the device at your desktop. In order to do so, you need a set of tools that take your design from concept to silicon. Processing your design includes design capture (including HDL synthesis

and/or schematic entry), implementation, and verification.

Capturing your Design

The first step in the creation of any logic design is to capture the intended functionality in electronic format. While Xilinx has a rich history in the support of schematic capture programs, over the last six years Xilinx has been working with leading vendors in support of HDL design flows (synthesis and optimization). This investment is paying dividends to designers using Xilinx devices in terms of technology specific optimizations that enable the creation of high performance designs from VHDL or Verilog.

The Xilinx Foundation Series™ software includes synthesis capabilities from industry leader Synopsys® (FPGA Express™). To ensure the highest quality of results through the synthesis process, Xilinx works cooperatively with all synthesis partners to develop proprietary optimization technology for Xilinx devices. This technology takes advantage of the extensive knowledge that the Xilinx R&D staff have of its silicon and implementation tools. These improvements are developed in a manner which allows them to be shared with third party synthesis vendors to ensure high quality results, through HDL design flows, regardless of your chosen synthesis provider.

Xilinx also delivers the industry's best design reuse methodology, enabling the seamless integration of intellectual property from either third parties (Alliance cores) or Xilinx (LogiCores). The Xilinx CORE Generator™ enables the customization of the core's parameters at your desktop, creating the exact functionality that your design requires. The use of SmartIP™ technology in the creation of cores ensures predictable, high-performance, scalable functions. One of the most popular cores is the Xilinx PCI LogiCore, currently offered as both 32-bit/ 33-MHz and 64-bit/66-MHz interfaces.

Implementation

Xilinx patented the timing-driven place and route of FPGAs in the early 90s. This technology allows you to specify timing requirements when creating your design, and automatically optimize your design with these requirements in mind. Advanced integration, between the synthesis tools and the Xilinx implementation tools, enable the passing of timing requirements from synthesis to place and route, and circuit delay information from place and route back to the synthesis algorithms.

This closed loop methodology not only saves time, by making sure that the place and route tools are working to create a design that meets all of your system needs, it also provides immediate feedback when timing requirements are unrealistic, given the current design description. In fact, the flexible verification methodology that Xilinx provides ensures that you are given feedback on the feasibility and quality of your design throughout the design flow. This Checkpoint Verification™ process ensures that you are spending your time most efficiently. The process

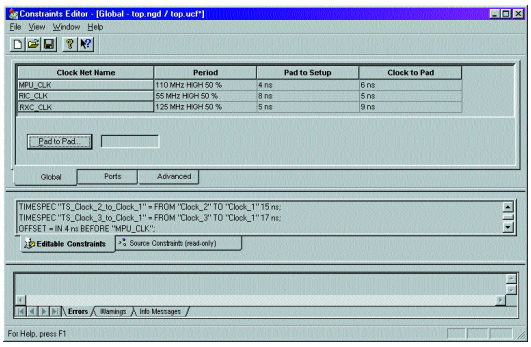


Figure 1: Constraints Editor

continues only with design iterations that will converge on your overall system requirements, and identifies trouble spots in designs that will not.

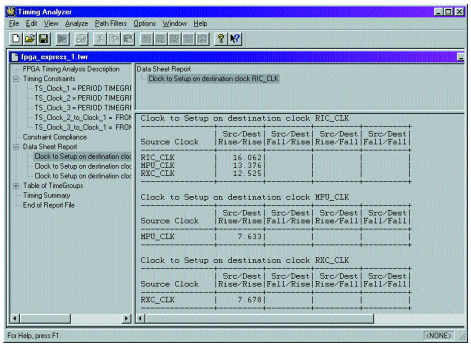


Figure 2: Training Analyzer

Xilinx has also made great strides in simplifying the process of timing driven design. The Xilinx "Constraints Editor" (Figure 1) guides you in your selection of signals and nodes to which you will apply timing constraints within your design. The Signal and Node lists are based on the same names you provided during the creation of the schematic or HDL source.

This easy-to-use GUI provides an intuitive interface for using industry's most comprehensive timing specification language—TimeSpecs™. Once these timing requirements are specified and the design is implemented, the Xilinx Timing Analyzer (Figure 2) is used to analyze the performance of a design. This intuitive interface streamlines the process of evaluating your designs timing by enabling a hierarchical analysis of all of the design's timing paths through the use of (+) expand and (-) collapse functionality.

The key to many design flows is the ability to rapidly iterate between design capture, implementation, and simulation (or in-system test). Iterative design is one key advantage of using programmable logic in your system, and to best take advantage of this, fast compilation times are important. To this end, Xilinx has dramatically reduced the place and route compilation times for its Spartan-II and Virtex families. Where traditional FPGA design flows implement designs at approximately 10K gates per minute, Xilinx v2.1i tools compile designs at a rate of approximately 100K gates per minute. For Spartan-II devices this translates into place and route times as fast as 1 minute, with average run times less than 10 minutes.

Verification

The increased density and performance of programmable logic is resulting in its frequent use as the central component in equipment design. As such, the emphasis on verification of programmable logic designs is becoming paramount to a program's success. Xilinx Check Point Verification methodology provides all of the hooks necessary to verify the operation of your design within the FPGA and within the target system. Elements of the Xilinx checkpoint verification model include:

- Support for functional, gate, and timing simulation.
- LMG SmartModel™ support for the Xilinx FPGA.
- Chip-level static timing analysis.
- STAMP™ model generation for board-level static timing analysis.
- In-System debug with Probe™.

Through this comprehensive suite of verification tools and data files, Xilinx provides you with the ability to employ the verification methodology of your choice. Our close relationships with

our Alliance EDA partners ensure success in the use of partner tools, while Xilinx also offers the Foundation Series solutions as a complete, ready to use package of EDA and place and route tools, automating both design compilation and verification.

Improving the Art of Logic Design

The Xilinx design methodology leverages the technological advantages of programmable logic, including the fact that all devices shipped by Xilinx are 100% functionally verified at the factory. This fact alone translates into weeks of savings in design time, as the processes of scan insertion and the re-verification of a design after scan insertion is not required.

Another key benefit in the Xilinx design flow is that the device is ”fabricated“ (programmed) by you, at your desktop, or in your manufacturing organization. Your retention of control of this process means that there is no Non-Recurring Engineering (NRE) cost associated with the creation of the device. This can dramatically reduce the emphasis frequently placed on running comprehensive (and time consuming) timing simulations after layout, and before device ”fabricated“.

For programmable logic, a relatively comprehensive verification regimen can consist of functional verification and static timing analysis if good synchronous design practices are followed. The streamlined Xilinx design flow reduces the necessity of timing simulation, which is frequently a time consuming process. If your environment calls for the verification of your chip design within the board (or system), Xilinx also generates the necessary timing information for use with board-level static timing analysis or simulation tools.

The ability to program the Xilinx device at your desktop also means that you can quickly iterate your design in a ”burn and learn“ fashion, reducing the pressure to get it right the first time. Problems discovered during in-system verification can actually be remedied in the chosen logic device. This is particularly useful when industry specifications or marketing requirements haven’t stabilized prior to the beginning of your project.

The combination of these factors means that the design methodology that you follow is streamlined when compared to the process required to design with an ASIC. The time saved in the design flow and system verification is frequently dramatic.

Conclusion

The advantages of creating your design with programmable logic include more efficient use of your time, faster time to market, and no NRE costs. Our experience has found these benefits frequently result in improved market success of our customers’ products. In addition to these benefits, the development systems required to design a Spartan device into your product are configured and priced to comfortably fit within any budget. Visit The Silicon Expresso Cafe (the Xilinx on-line store) or contact your local Xilinx sales representative to learn more.

See page 63 for an overview of our software products.

How To Create an MP3 Player Using Spartan-II FPGAs.

Spartan-II FPGAs are used to implement complex MP3 system-level glue logic.

by Jasbinder Bhoot, Manager, Strategic Applications, Xilinx, jasbinder.bhoot@xilinx.com

MP3 is rapidly becoming the defacto standard for the delivery of high quality music on the Internet. This technology has been well received as evidenced by the over five million MP3 software plug ins that have been downloaded to date. This is a testament to the future potential of the MP3 technology considering the relatively limited marketing it has received to date.

MP3 is an abbreviation for MPEG 1 layer 3, which is a compressed digital audio format that keeps the file size small without losing the quality of the original audio. MP3 allows audio files to be compressed to approximately 1/11th of the original size. For example, a typical music CD consumes 650MB; using MP3 the same audio only takes 55MB! This has allowed music to be stored on the PCs hard drive, allowing users to effortlessly create and customize music play lists.

The compression achieved by MP3 makes it practical to construct a solid state portable audio player based upon FLASH memory as the storage medium. This article explores the development of a portable MP3 player using the Spartan-II FPGA family.

Design Overview

The MP3 player discussed in this application contains advanced user interface features, such as the ability to store contact information, record memos, and other functions typically found in Personal Digital Assistants (PDAs). The design uses an IDT RC32364 RISC processor to decode the MP3 data and implement the graphical user interface. The Xilinx Spartan-II FPGA is used to implement the complex MP3 system-level glue logic required to interface and manage the memory and I/O devices.

Figure 1 shows a block diagram of the design. The key features are:

- 128 x 128 pixel graphical touch screen.
- USB interface for music downloads and network connectivity.
- IRDA-compliant infrared interface for exchanging data with other units.
- 32 MB of on board FLASH storage.

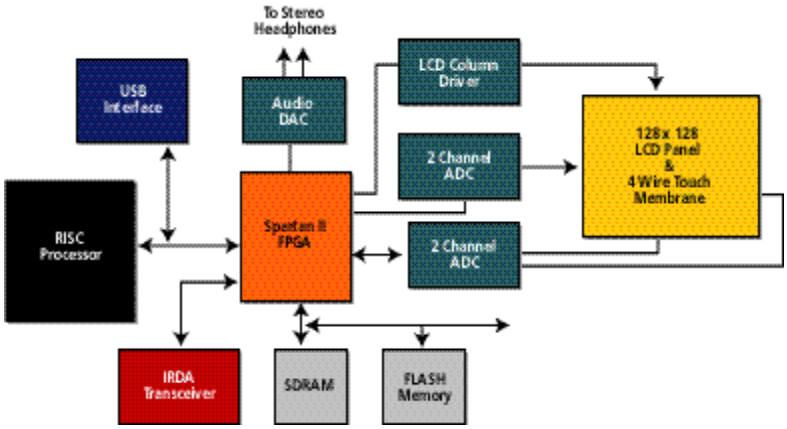


Figure 1: MP3 system block diagram.

- CompactFlash interface for storage expansion using CompactFlash cards or MicroDrive hard drives.

Use of ASSPs

The design uses Application Specific Standard Products (ASSPs) to implement much of the complex logic. Typically, these ASSPs are not designed to communicate with each other. The Xilinx Spartan-II FPGA is used to provide the complex glue logic for the interface between the ASSPs and the RC32364 RISC processor from IDT.

The Digital-to-Analog converter is the Crystal CS4343 from Cirrus Logic. The CS4343 provides the analog stereo headphone interface; a serial port is used to transfer digital audio data streams, while an I^2C control port is used to configure the device features such as volume, muting, equalization, and power management.

The USB interface is the USBN9602 from National Semiconductor. This device supports full speed USB and includes an integrated USB transceiver. The system interface for the USBN9602 is an 8-bit microprocessor bus that can be configured to operate in a multiplexed or non-multiplexed mode. To reduce the number of pins, the multiplexed mode was used.

The FLASH memory is the Samsung KM29U64000T 8M x 8 device based on NAND FLASH technology. This memory is very popular in MP3 players due to its high density and low cost per bit. However, this FLASH memory contains two characteristics that present significant design challenges:

- The KM29U64000T uses a highly multiplexed 8-bit-wide port for both address and data access.
- Error detection and correction is needed to ensure system integrity.

The second and most challenging issue, data integrity, is common with NAND-based FLASH technology. The FLASH memory contains a range of valid memory blocks (N_{VB}). For the KM29U64000T the typical N_{VB} is 1020, the minimum is 1014, and the maximum is 1024. While the first block is guaranteed to be good, bad blocks can occur at any other location within the memory array. Invalid blocks are marked at the factory by storing a “0” value at location “0” in either the first or second block of the page. The design must keep a record of good blocks resulting in a non-contiguous memory map. A further issue is that the FLASH memory may experience additional block failures during the memories operational life.

Glue Logic Architecture

Figure 2 shows an overview of the architecture implemented in the Spartan-II FPGA.

The architecture consists of the following functional blocks:

- IP Bus Controller.
- CPU interface.
- LCD controller.
- Memory Interface.
- SDRAM controller.

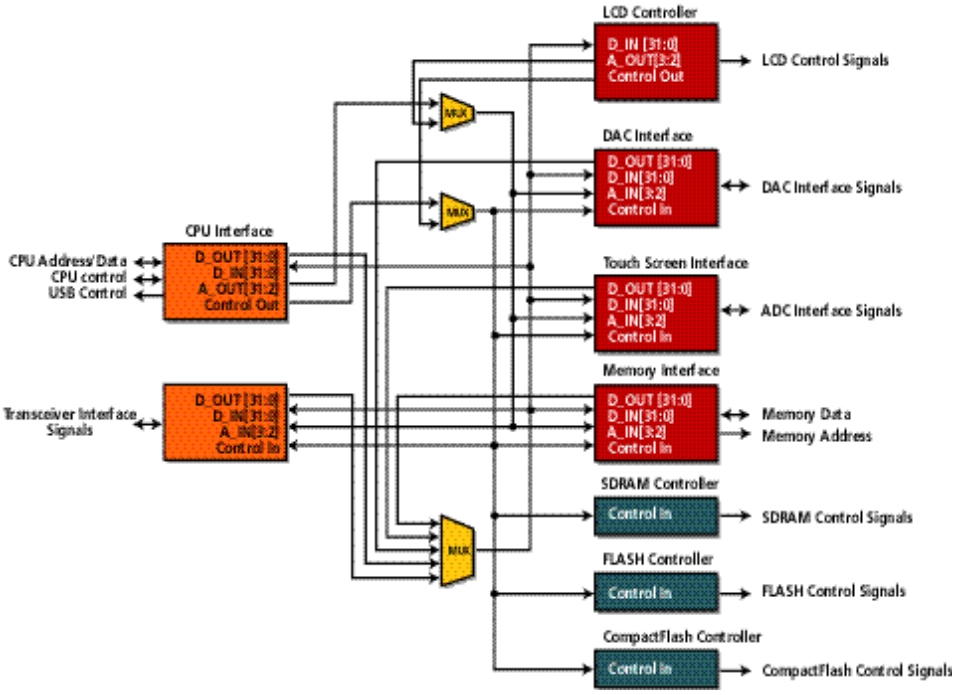


Figure 2: Spartan-II block diagram.

- FLASH Controller.
- CompactFlash Controller.
- IRDA Controller.
- Audio DAC Interface.
- Touch Screen Interface.

A simple non-multiplexed, multi-master address data bus called the IP bus, interconnects the blocks. Having the FLASH, SDRAM, and the CompactFlash RAM share a common address and data bus allows a reduction in pin count.

The IP Bus has two masters, the CPU interface and the LCD controller. Multiplexers are used for gating data into the internal datapaths, eliminating the need for 3-state drivers.

The CPU Interface performs the CPU initialization, the protocol conversion (to and from the CPU bus, USB interface, and the IP bus), and the address de-multiplexing.

The LCD Controller is responsible for refreshing the screen with the image stored in the SDRAM. The LCD controller can obtain the data for screen refresh independent of the CPU activities.

The Memory Interface block implements the data path required to map the 8- and 16-bit memory devices to the 32-bit IP bus. Although the RC32364 is capable of obtaining instructions and data from devices with varying bus widths, using the Spartan-II to implement this function reduces the CPU bus cycles and hence, increases performance and reduces power consumption.

The SDRAM controller is based on the design developed by Xilinx in application note XAPP134: Virtex Synthesizable High Performance SDRAM Controller There are two changes made to this design:

- The host interface is adapted from a multiplexed address data bus to a non-multiplexed data bus.
- A 16-bit wide SDRAM memory configuration is needed in place of the 32-bit wide memory datapath.

The FLASH Controller copies the executable image from the FLASH memory to the SDRAM at boot time, to overcome the FLASH random access latency and maximize system performance. This method also allows the NAND FLASH error code correction to be implemented in software, resulting in an efficient use of the FLASH memory.

The CompactFlash Controller provides the interface to allow the MP3 music files to be stored in to the CompactFlash memory via the USB serial link. The control signals required to retrieve the MP3 music file for playback are also contained in this block.

The IRDA Controller is essentially a specialized, fixed function UART. Separate, 2-word receive and transmit FIFOs are used to reduce the interrupt overhead associated with data transmission. The IR transceiver can support a data rate of 115 Kb/s resulting in a CPU interrupt every 557 ms.

The Audio DAC Interface provides the dedicated hardware needed to implement the transfer protocol for delivering an uninterrupted audio stream. This hardware consists of two, 4-word FIFOs, one for each audio channel and a state machine to manage the FIFOs and sequence the interface signals. The audio DAC interface also contains a 2-bit I/O port that uses software to implement the I^2C protocol used for accessing the control and status registers in the DAC.

The Touch Screen Interface is an I/O port that allows the processor to read the data returned by the two-channel analog-to-digital converter. This allows the system software to determine the X and Y touch screen coordinates.

Spartan Device Selection

Spartan-II devices are available in a wide range of densities and packages. The following criteria were used to select the device:

- I/O Pins. The design requires a total of 137 I/O pins.
- Voltage. The design operates at 3.3V.
- Density. The estimated size of the design is 83,000 gates.
- Performance. The highest clock speed used in this application is 64 MHz; this is used to clock the SDRAM controller. The remaining logic operates at sub multiples of this clock.
- Packaging. The size constraints imposed on most modern designs dictates a high-density surface mount package.

Based on these criteria the device selected for this design is the XC2S100. This device offers 100K gate density, 3.3V operation, 176 user I/Os, and is packaged in a FG256 BGA package. The cost of this device, in 100K quantities, is $12.95.

Conclusion

This design illustrates how Spartan-II FPGAs can be used to provide time-to-market advantages in high volume consumer applications. In this application, the Spartan-II FPGA is used as a cost-effective device to maximize the CPU performance and reduce system power consumption by providing some dedicated functions as well as the glue logic required to interface to the ASSPs. Using a Spartan-II FPGA also allows field upgrade flexibility, in a market where standards and protocols are still evolving.

Inverse Multiplexing for ATM (IMA)

A Programmable ASSP Solution for transmitting high-bandwidth data over multiple T1 or E1 lines.

by James D. Beatty, President, Applied Telecom, jim@apptel.com, and Krishna Rangasayee, Manager, Strategic Applications, Xilinx, krishna@xilinx.com

The Spartan-II Family, combined with an extensive soft intellectual property (IP) portfolio is the first programmable logic solution to effectively penetrate the ASSP marketplace. The ATM IMA-8 core from Applied Telecom, ported to the Spartan XC2S150 device, is a good example, highlighting the concept of a programmable ASSP.

Applied Telecom is the newest member of the Xilinx AllianceCORE program and brings a wealth of expertise in ATM, SONET, telecommunications, and networking applications. The IMA-8 core, developed, sold, and supported by Applied Telecom, targets network access systems such as adapters, multiplexers, and switches. Several leading manufacturers, including Alcatel, Ericsson, Nokia, and Nortel, are already using Applied Telecom's Xilinx-based IMA technology in production systems.

What is IMA?

IMA stands for "Inverse Multiplexing for Asynchronous Transfer Mode" (ATM) and it allows the transmission of a high-bandwidth stream of ATM cells over multiple T1 (1.544 Mbps) or E1 (2.048 Mbps) facilities (or circuits). IMA is applicable to both public and private networks and allows end users to enjoy the many benefits of ATM, such as Quality of Service (QoS) provisioning, scalability, and the ability to easily mix data, voice, and video. IMA circumvents the high cost and unavailability of broadband transmission facilities such as T3, E3, and SONET/SDH by using only as many lower cost, lower bandwidth facilities as necessary. With the advent of Digital Subscriber Line (DSL) technology, the case for IMA is even greater.

IMA Applications

IMA is applicable to many different types of ATM Wide Area Network (WAN) access equipment including ATM switches and routers with WAN ports, ATM access concentrators and multiplexers, and communications servers with WAN NICs. Typically, IMA is used as the WAN interface for general purpose access multiplexers, traffic aggregators, and access switches. Another common application is in Digital Subscriber Line Access Multiplexers (DSLAMs) where IMA can be used either to interconnect the DSLAM with a Remote Access Multiplexer (RAM) or as the high speed network-side interface. Emerging applications are extending the IMA protocol beyond T1 or E1 to include other facilities using DSL circuits.

IMA Operation

Figure 1 illustrates the basic IMA mechanism for sending a single ATM cell stream over a number of lower speed transmission facilities or links.

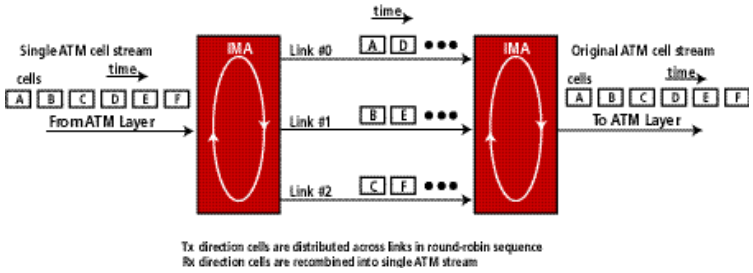


Figure 1 - Simplified IMA process.

Layers	Sub-layer	Functions
ATM Layer		
Physical (PHY) Layer	IMA Specific Transmission Convergence Sublayer	ATM cell stream splitting and reconstruction
		Differential delay accommodation
		IMA Control Protocol (ICP) cell insertion / removal
		Cell rate decoupling
		IMA frame synchronization
		Cell stuffing,asynchronous facility compensation
	Interface Specific Transmission Convergence (TC) Sublayer	Discard cells with HEC errors
		Header error correction
		HEC generation / verification
		Cell scrambling / descrambling
		Cell delineation
		Scrambling / descrambling
	Physical Media Dependent (PMD) Sublayer	Transmission frame generation / recovery
		Bit timing,line coding
		Physical medium

Table 1 - IMA sublayer reference model.

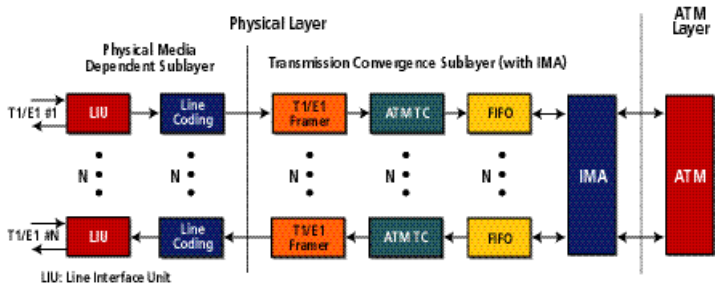


Figure 2 - Typical PHY implementation.

When splitting an ATM virtual circuit among multiple T1 or E1 links, the IMA subsystem must insert the IMA-specific cells into the transmitted ATM cell streams. In the receive direction, a cell-based IMA framing process is used to locate and remove the IMA-specific cells so that only the ATM cells are passed to the ATM Layer.

A variable number of physical links and ATM bandwidth rates can be supported and mechanisms are specified for accommodating differential delay variations present in the transmission links and for handling link failures and changes to the available transmission bandwidth.

Layer Reference Model

In terms of the protocol layer reference model, the IMA sublayer is considered to be an extension of the Physical Layer (PHY), sitting below the ATM Layer (ATM) and, as much as possible, transparent to the ATM Layer device. Table 1 illustrates the functions performed by the IMA sublayer.

From an ATM layer perspective, the behaviors exhibited by IMA groups are different than those of real PHY facilities. Two examples include the effects of IMA group start-up and variations in bandwidth caused by the activation/deactivation and addition/deletion of links.

In addition to the PHY layer, IMA also affects the management layer. The management layer handles alarm detection and processing, making it possible to configure IMA groups, add and delete links, and maintain IMA sublayer statistics.

PHY Layer Considerations

A typical PHY layer implementation with IMA is shown in Figure 2. In many such implementations, the Physical Layer function is resident on a "line card" which is physically separate from the ATM layer device to allow the ATM device to serve many facilities. This co-location of IMA on the line card restricts the facilities which can be allocated to an IMA group because only the specific T1 or E1 facilities attached to that line card can be grouped, limiting the configurability of the system.

One solution, for maximum configurability, is to place the IMA function with the ATM layer device. But this is usually difficult to implement and causes some system-level functional partitioning problems (such as splitting up the PHY layer across modules). A more common solution that sacrifices some of the flexibility in assigning facilities to IMA groups is to develop the IMA functionality and the line card so that each T1/E1 facility can be independently configured to be part of an IMA group or be bypassed around the IMA function to be accessed uniquely by the ATM layer device. With this solution, the line card is no longer a "dedicated" IMA card.

IMA Solution

Given the availability of many off-the-shelf devices providing the T1/E1 line interface, framing, and ATM Transmission Convergence (TC) sublayer functionality plus the wide acceptance and usage of the ATM Forum's UTOPIA bus interface for ATM cell transfer, it is natural to define an IMA solution that can be inserted between the TC function and ATM layer. With the availability of the Spartan-II family, a complete IMA solution can be implemented using a single XC2S150 device, an external SRAM device, and a software driver.

The IMA-8 Core

The IMA-8 product is an XC2S150 device solution that supports up to eight links and four IMA groups. The external interfaces for this FPGA device are shown in Figure 3. The IMA

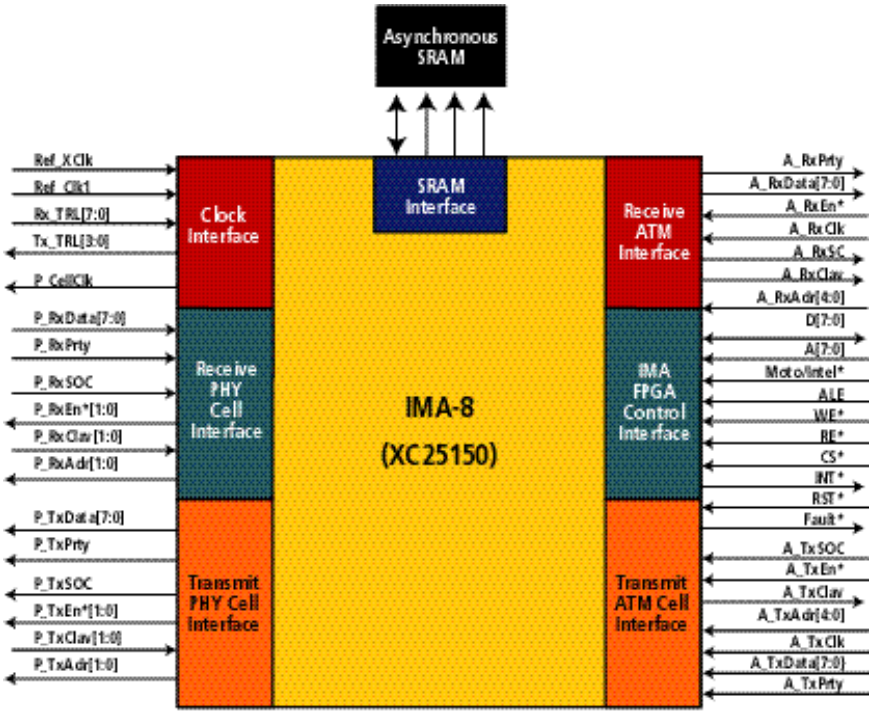


Figure 3 - IMA-8 interfaces.

implementation has been partitioned so the real-time processes are performed in the FPGA and all non-real time processes are performed by a software driver. For example, all IMA link state machines are implemented in the FPGA but the IMA group state machines are implemented in soft-ware. This partitioning eliminates interrupts from the FPGA and allows the software to operate as a periodic background task on the processor.

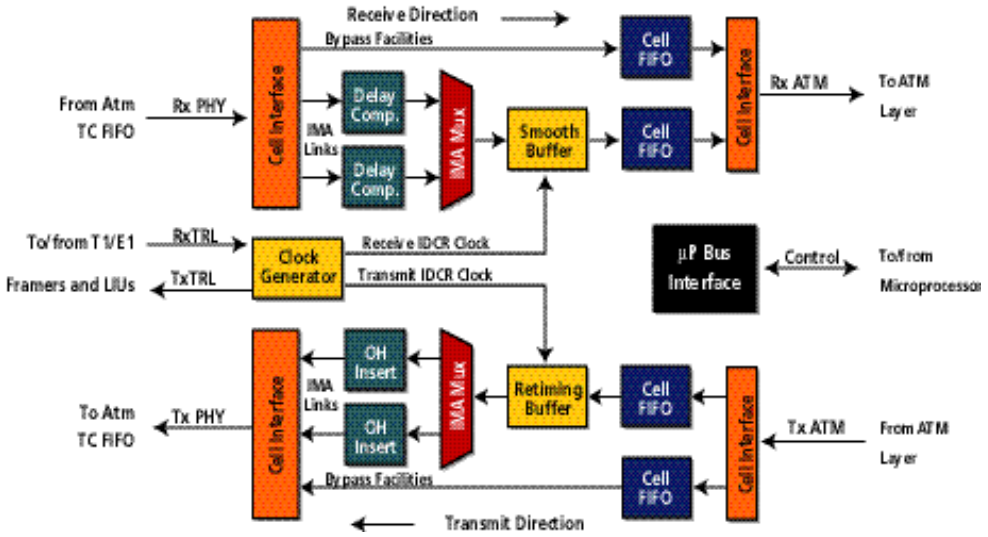


Figure 4 - Functional block diagram.

Functional Description

A simplified block diagram of the IMA FPGA solution is shown in Figure 4. The solution is composed of four main functional areas:

1 - the IMA clock generators.

2 - the Transmit IMA processing.

3 - the Receive IMA processing.

4 - the Microprocessor Bus Interface.

An IMA software driver completes the IMA implementation. To get specific details on these components, please contact Applied Telecom or the Xilinx High Volume Business Unit.

Conclusion

Early deployment of IMA technology meeting the IMA v1.0 standard began in late 1997, but due to different interpretations of this specification, true multi-vendor interoperability was not really possible until the completion and acceptance of the IMA v1.1 specification in 1999. Throughout this period, the changes to the applicable technical standard and the lessons learned through inter-vendor testing required the flexibility of FPGA and software implementations.

At present, stability in standardization plus large scale IMA deployment have set the stage for the introduction of standard silicon IMA products. But IMA is still an emerging technology with limited test equipment support and compliance test suites. An FPGA based IMA solution with efficient partitioning of hardware and software functions provides the necessary scalability and flexibility to handle all of these applications and allow for tracking of new standards.

With the introduction of the Spartan-II FPGA family and the IMA8L core (along with other Xilinx-based IMA core solutions) an FPGA based IMA implementation is simple and economical. Σ

The IMA-8 core is available immediately for use in Spartan-II FPGAs. An evaluation board and the DRV-IMA software are also available now. All IMA products can be purchased directly from Applied Telecom.See www.apptel.com.

HDLC Controller Solutions Using Spartan-II FPGAs

HDLC Controller cores, ported to the Spartan-II family, highlight the concept of a programmable ASSP

by Amit Dhir, Sr. Engineer, Strategic Applications, Xilinx,
amitd@xilinx.com

HDLC Controller cores (soft IP) have been available for Xilinx XC4000XL and Virtex FPGAs for some time. Now, this technology is also available for use with the new Spartan-II family, which is uniquely poised to penetrate the ASSP marketplace because of its advanced features, high performance, and low cost. A programmable HDLC Controller solution, with efficient partitioning of hardware and software functions, provides the necessary scalability and flexibility you need for any application, and allows you to easily adapt to new standards; you get all the benefits of an ASSP device, plus the many advantages offered by programmable logic.

HDLC stands for "High-Level Data Link Control," a bit-oriented synchronous data link layer protocol developed by the International Standards Organization (ISO). HDLC Controllers are devices which execute the HDLC protocol and their properties include:

- Transmitting and receiving the serial packet data.
- Providing data transparency through zero insertion and deletion.
- Generating and detecting flags that indicate HDLC status.
- Providing 16-/32-bit CRC on data packets using the CCITT defined polynomial.
- Recognizing the single byte address in the received frame.

HDLC Controller Applications

HDLC Controllers are used in various data networking operations. Some of the key applications are:

- Frame relay switches - high density access, FRADs.
- ISDN - basic-rate or primary-rate interfaces, D-channel.
- X.25 and V.35 protocols.
- Internet/edge routers, bridges, and switches for high bandwidth WAN links.
- Cellular base station switch controllers.
- Error-correction in modems.
- T1/E1, T3/E3 - channelized, clear channel (unchannelized).
- xDSL - each port can support up to 10Mbps.
- Dual HSSI.
- SONET termination.
- Digital sets, PBXs, and private packet networks.
- C-channel controller to Digital Network Interface Circuits.
- Data link controllers and protocol generators.

- Inter-processor communication.
- Logic consolidation.
- CSU/DSU.
- Protocol converters.
- Packet data switches.
- Distributed packet-based communications systems.
- Multiplexer/Concentrators - remote access, multi-service access.

Xilinx AllianceCORE Partners

Currently, there are two Xilinx AllianceCORE partners supplying HDLC cores for Spartan-II FPGAs: Memec Design Services, and CoreEL MicroSystems.

Memec Design Services (MDS)

The Single Channel XF-HDLC Controller core conforms to the ISO/IEC 3309 specification, and provides the entire functionality of the HDLC Controller. The core:

- Provides 16-/32-bit CCITT-CRC generation and checking.
- Performs flag and zero insertion and detection.
- Allows full duplex operation.
- Operates at a DC to 53 Mbps (STS-1) data rate.
- Provides full synchronous operation.
- Provides an interface that can be customized for user FIFO and DMA requirements.
- Allows 16-/32-bit FCS generation and verification.
- Provides an MTU and compression enable signal.
- Provides a scramble and de-scramble enable signal.
- Detects the Address field, Control field, escape sequence, and FCS packet errors.
- Provides statistics for Address field, Control field, Protocol field, FCS field, and escape sequence packet errors.
- Provides statistics such as the number of packets, runt packets, valid packets, and excess length packets.
- Detects error conditions like Transmission Break on transmit side.
- Discards packets received with Address, Control, or Protocol field errors.
- Optionally compresses Address, Control, and Protocol fields.
- Generates a discard packet signal for any

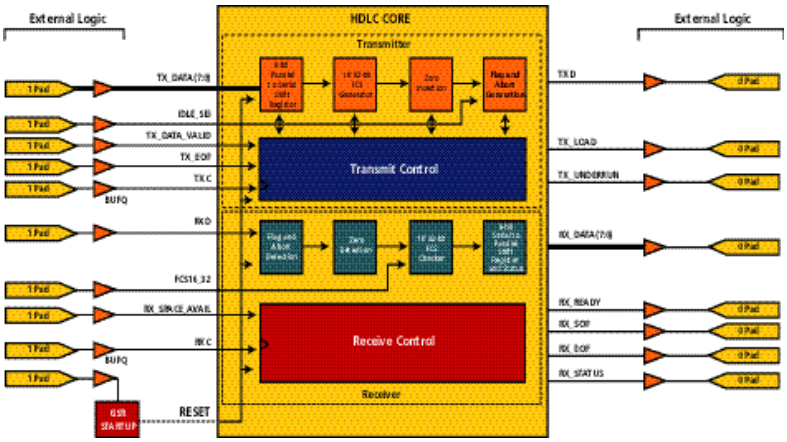


Figure 1 - HDLC Controller block diagram.
(courtesy: Memec Design Services)

CoreEL MicroSystems

The PPP8 HDLC core (CC318f) conforms to RFC1619 PPP over SONET specification. The core:

- Supports programmable Address, Control and Protocol fields.
- Supports an 8-bit Packet and PHY framer interface.

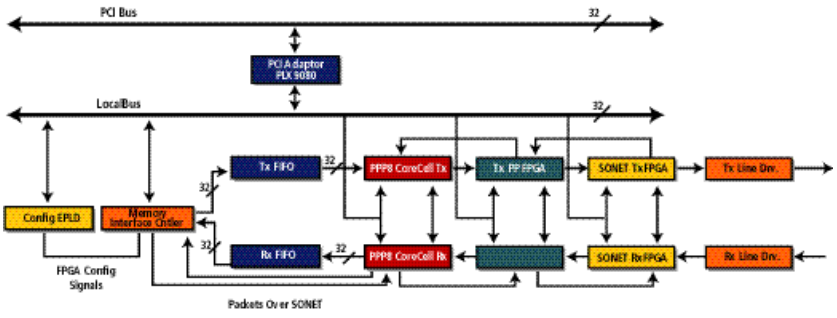


Figure 2 - Application of HDLC cores.
(courtesy: CoreEL MicroSystems)

packet with an FCS or invalid packet on packet interface error.

A typical HDLC Controller Block Diagram is shown in Figure 1. A typical application is shown in Figure 2.

Comparing the Spartan-II HDLC Solution with Stand-alone ASSPs

Using an HDLC IP core in conjunction with a Spartan-II FPGA, offers significant advantages over a fixed-logic (non-programmable) ASSP. By using the Spartan-II family, you can implement the feature sets required and these can be customized to meet the exact design requirements.

You can also integrate other parts of your design within the same FPGA, for increased performance and reduced cost. For example, an HDLC Controller supplied by a stand-alone ASSP vendor may include a 32-bit, 33-MHz PCI Controller. However, your design may require something different, such as a 32-bit, 66-MHz or a 64-bit, 33-MHz PCI Controller. By using the Spartan-II FPGA family in conjunction with the appropriate cores, you can create the optimal HDLC to PCI solution for your specific requirements. Because of the inherent low cost of the Spartan-II family, this flexibility comes at a significantly lower cost than the fixed ASSP solution. Figure 3 shows the value comparison.

Programmability is Key

Conflicting specifications and lack of a clear direction create the need for programmableASSP solutions. The Spartan-II family accommodates specification changes and can easily be used in volume production. For example, the currently available non-programmable HDLC Controller solutions are based on X.25 (CCITT) level-2 or OSI Layer-2, ISO3309 specifications only. However, the solutions provided through Spartan-II FPGAs allow you to use X.25 (CCITT) level-2, OSI Layer-2, ISO3309, RFC1619 PPP over SONET, and ITU recommendation (Q.921) specification standards. It would be nearly impossible, and cost-prohibitive, for an ASSP vendor to meet all of these specifications.

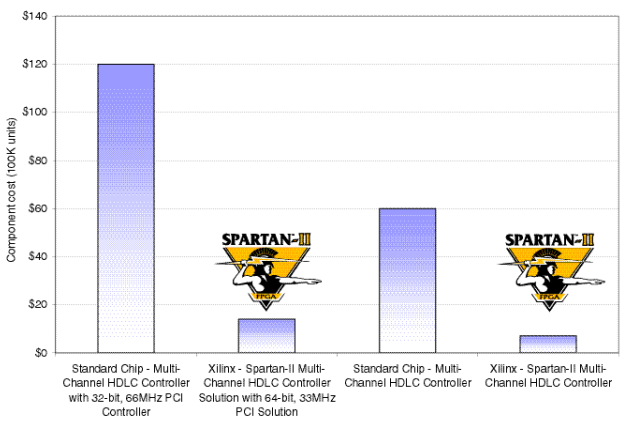


Figure 3 - Value comparison.

Xilinx Online for Field Updates

Through the Xilinx On-line program, the Spartan-II family allows you to remotely update your design, over any network. With new features, enhancements, and bug fixes, the life of the HDLC controller within any networking system increases. This also allows your equipment to adapt to changing standards. Designing systems that allow remote upgrades can provide new revenue opportunities as well, because you can continue to sell new features, after your equipment is installed. You cannot easily offer these unique features if your hardware design is not programmable.

Conclusion

The Spartan-II family is unaffected by the hurdles that an ASSP vendor must overcome; its inherent advantages extend the reach of the Spartan family to new levels and creates new opportunities for PLDs in the ASSP market.

The cost difference between a stand-alone ASSP and an equivalent programmable Spartan-II solution is considerable. Thus, the Spartan-II family is a clear winner in not only the HDLC Controller market, but also other ASSP niche areas.

For more information on the MDS core, see:
www.xilinx.com/products/logicore/alliance/memec/memec.htm.
For more information on the CoreEL MicroSystems core, see:
www.xilinx.com/products/logicore/alliance/coreel/coreel.htm.

Low Power Benefits of the Spartan-XL Family

A look at Spartan-XL power-down modes, small form factor packages, package power dissipation, and device reliability

by Ashok Chotai, Senior Competitive Marketing Engineer, Xilinx, ashok@xilinx.com

The Spartan™-XL family is built with an advanced 3.3V process, a segmented low power architecture, and a power-down feature that significantly reduces the FPGA's quiescent current requirements (from 3mA to 100µA typical). This opens up a whole new market for Spartan-XL FPGAs because these devices can now be used in power-sensitive applications such as laptop computers, cellular phones, PDAs, camcorders, and so on.

Power-down Modes

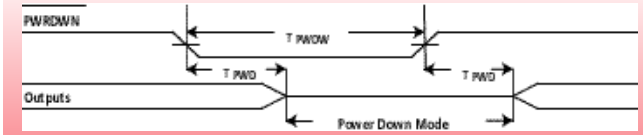
There are two kinds of power-down modes: manual and automatic. Both provide low quiescent (standby) current while retaining the bit map (configuration data with which the device had been configured).

Manual Mode

In the manual mode, the device is fully inactive, the register data is lost, and the activation and de-activation are controlled by the PWRDWN pin. The following events occur in sequence to conserve the power:

- All inputs (including M0, M1, DONE, CCLK, and TDO) except PWRDWN are disconnected from their sources. Internal to the device, the input signals are tied to GND.
- All pull-up and pull-down resistors on all I/Os (except PWRDWN) are disabled.
- The Global Set-Reset (GSR) is activated, clearing all the registers in the device. This reset state is held as long as the device is in power-down mode.
- The Global 3-state (GTS) is activated, putting the device outputs in high-impedance state.

The device stays in DC state, drawing minimal power, until PWRDWN goes High, at which point it returns to full operation over a period of 50 ns (max), as shown in Figure 1. The manual power-down mode is described in detail in application note XAPP124 on the Xilinx website at: http://www.xilinx.com/xapp/ xapp124.pdf.



Description	Symbol	Min	Max
Power-down Time	T_{PWD}	-	50ns
Power-down Pulse Width	T_{PWDW}	50ns	-

Figure 1: Power-down timing.

Automatic Power-down Mode

In the automatic mode the device is active, the register data can be retained if required, and the power-down is initiated without using the PWRDWN pin. You can selectively control the features of a design, which may consume a relatively large amount of power, to obtain quiescent (standby) current down to 100µA typical. Some of these controllable features are:

- Pull up and pull down resistors on the IO pins.
- 5V I/O tolerance.
- Clocks that need to run intermittently.
- The redundant use of high-frequency clocks.

The critical register data can be left operating, while disabling the remaining parts of the design, to obtain the low quiescent power. Unlike the manual power-down mode, this mode does not have any power-down recovery time when switching back to normal operation. The automatic power-down mode is described in detail in the application note XAPP125 on the Xilinx website at: http://www.xilinx.com/xapp/xapp125.pdf.

Superior Benefits

The low power requirements of the Spartan-XL family makes it possible to use small form factor packages to save board space and reduce design costs, making it an ideal choice for most portable and low power equipment.

Package Power Dissipation Limit

Power consumption plays an important role in selecting the device package. For the device to operate reliably, the power consumed by the device must be less than the maximum power its package can dissipate. Use the following equation to determine the maximum power dissipation P_d for a particular package:

$$P_d = (T_J - T_A)/\theta_{JA}$$

where T_J is the maximum junction temperature, T_A is the ambient temperature, and θ_{JA} is thermal resistance of the package.

Lower power means lower heat dissipation requirements, thus making it possible to use smaller footprint packages. This in turn would provide cost savings because the package does not need to have an extra heat sink, and it occupies less board space. As an illustration of this power-performance limit, the graph shown in Figure 2 plots dynamic power with respect to performance for two devices of comparable logic density: the Xilinx XCS30XL in 144-pin Chip Scale package and the Altera 10K30A in 144-pin TQFP package.

Figure 2 shows that the Chip Scale package in the Spartan-XL family can be used up to 100 MHz without any problem with power dissipation. However, the TQ package used for the Altera 10K30A device cannot be used above 77 MHz. The device is just not reliable beyond this performance. To use it reliably beyond 77 MHz performance, you would need to either add a heat sink or force air into the package using a

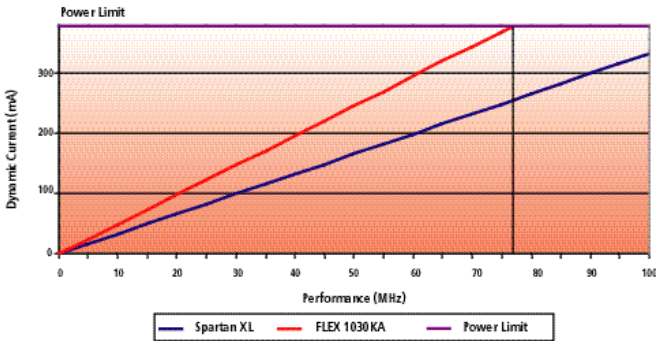


Figure 2: Dynamic power comparison of Spartan-XL device with 10K30A.

fan, which means extra hardware, additional cost, and more board space.

Small Form Factor Advantage

The Chip Scale package (CSP) dramatically reduces board space and increases I/O count; it is smaller than any competitive offering in the industry. The package is available in 144 and 280 ball counts with 0.8 mm pitch, and is ideal for low-power, light-weight, and small form factor designs. Xilinx is the first programmable logic supplier to offer the CSP package for an FPGA that meets the JEDEC Level 3 moisture sensitivity level requirements. This level of reliability enables you to reduce standard manufacturing cycle times and further minimize overall system cost.

As seen in Figure 3, the CS144 package offers 70% board area savings when compared with the TQ144 package. Similarly, the CS280 package offers 83% board area savings when compared with the popular PQ240 package.

Higher Reliability

Xilinx is known for quality and reliability. Reliability is measured in terms of Failure-In-Time rates (FIT); failures in 10^9 device hours. Lower power provides lower FIT rates and higher device reliability, thus reducing product test rework and field failures. Overall device reliability decreases exponentially as junction temperature increases. The industry standard limit for the maximum junction temperature is 125°C for the plastic package and 150°C for the ceramic package. Table 1 compares FIT rates of Xilinx to that of the nearest competitor.

T_J (°C)	50	60	70	80	90	100
Altera	7	20	50	118	267	578
Xilinx	1.5	4	10	23	53	115

Table 1 - FIT rate comparison.

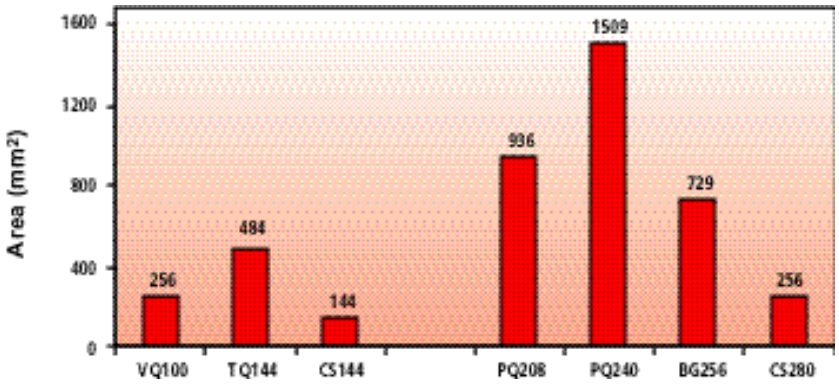


Figure 3: Package area comparison.

Conclusion

The low power requirements of the Spartan-XL family makes it possible to use small form factor packages to save board space and reduce design costs, making it an ideal choice for most portable and low power equipment.

The New XPLA3 CPLD Family

The Best CoolRunner Family Yet

CoolRunner devices are ideal for low-power, high-perfor mance applications.

by Reno Sanchez, CoolRunner Marketing & Applications Manager, Xilinx, renos@Xilinx.com

The CoolRunner™ CPLD families are the world's only CPLDs using the patented Fast Zero Power (FZP) design technique to simultaneously deliver high performance and low power consumption. These devices offer pin-to-pin (TPD) delays of 5.0 ns (or greater than 250 MHz system operation), and less than 100 uA of standby current (approximately 1/3 of the power consumed by all other competing CPLDs at FMAX).

These characteristics make CoolRunner devices ideal for low power applications that include portable, handheld, and power-sensitive applications. These devices are also idea for systems that have a strict power or thermal budget, a need for increased system reliability (less power means less activation energy and lower FIT rates), a need for lower cost (by eliminating or reducing the system cooling requirements), or a need for reduced power supply requirements (or battery operation). Figure 1 illustrates the CoolRunner CPLD families.

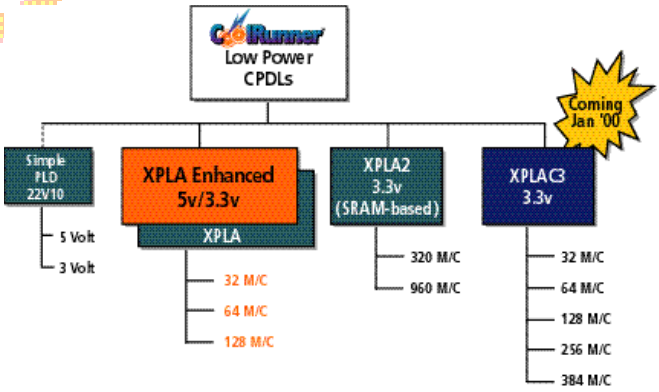


Figure 1 - CoolRunner CPLD family.

The XPLA3 Family

The XPLA3™ (eXtended Programmable Logic Array) is the newest CoolRunner CPLD family, and includes devices ranging from 32 to 384 macrocells. XPLA3 was created to maintain the same competitive advantages as the existing CoolRunner families, add additional features, and increase performance, while delivering all this at a substantially lower cost.

XPLA3 Architecture

The XPLA3 architecture is based on a PLA (Programmable Logic Array). The PLA is a programmable AND Array combined with a programmable OR Array. All other competing CPLDs employ a PAL (Programmable Array Logic) that combines a programmable AND Array with a fixed OR Array. Having a programmable OR Array allows product terms (PTs) to be shared between macrocells (effectively increasing design density because there is no duplication of logic), excellent pin locking (every PT is available

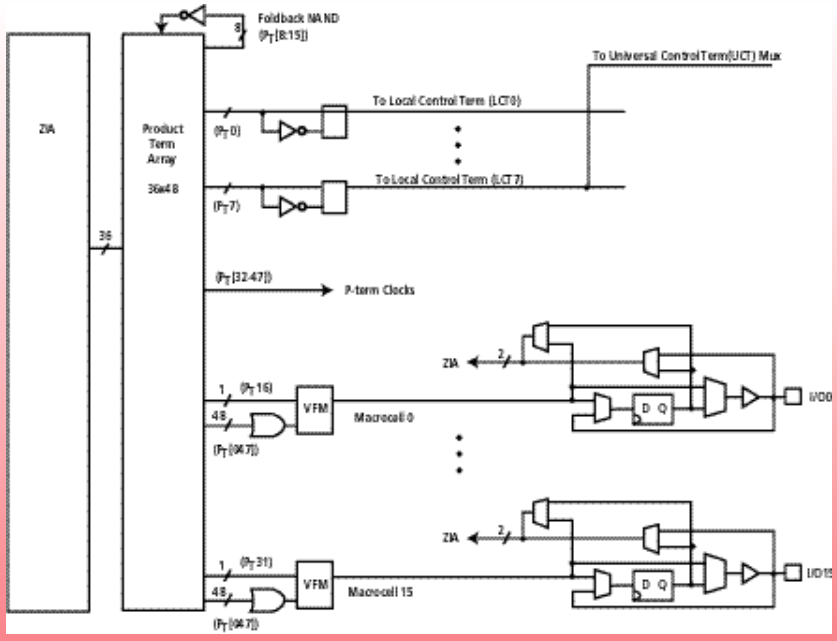


Figure 2 - XPLA3 Logic Block architecture.

to every macrocell), and very simple timing model (timing remains the same regardless of how many PTs are used).

The XPLA3 architecture includes a pool of 48 product terms that can be allocated to any output in the logic block, a direct input register path, multiple clocks (both dedicated and product term generated), and both reset and preset for each macrocell.

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, clock terms, PLA array, and 16 macrocells. There are 36 pairs of True and Complement inputs from the ZIA that feed the programmable OR array. In addition there are 8 foldback PTs that are available for ease of fitting and pin retention. Also within the 48 p-terms there are eight local control terms (LCT 0-7) available as control inputs to each macrocell for use as asynchronous clocks, resets, presets, and output enables. The other PTs serve as additional single inputs into each macrocell. Sixteen of these are coupled with the associated programmable OR gate into the VFM (Variable Function Multiplexer). The VFM increases logic optimization by implementing any of the OR, XOR, XNOR, or NOR functions before entering the macrocell.

Each macrocell can support combinatorial or registered inputs; a global preset and reset for each macrocell; and configurable D, T, or L registers, with maximum clocking flexibility. Each of these flip-flops can be clocked from any one of nine sources. There are two global synchronous clocks that are derived from the four external clock pins via a four-to-two multiplexer. There is also one universal clock signal sourced by a global control term. The clock input signals LCT4-LCT7 (Local Control Terms) can be individually configured as either a product term or sum term equation created from the 36 signals available inside the logic block.

Conclusion

The XPLA3 CoolRunner CPLD family simultaneously delivers both high performance and low power consumption at a very competitive price, and will be the lead CoolRunner CPLD family for Xilinx in 2000.

CoolRunner Power Estimator Tool

Here's how to determine how much power your design will use.

by John Hubbard, CPLD Applications Engineer, Xilinx,
john.hubbard@xilinx.com

The CoolRunner Power Estimator tool was developed to help you estimate the power consumed by your CoolRunner CPLD designs. This is an easy to use spreadsheet and Perl script available for download from the Xilinx website. The latest version is available for download from WebPACK™ by selecting the Utilities button.

Estimating Low Power

Estimating the power consumption of CoolRunner CPLDs is quick and easy. After you have targeted a particular device, you import the data created by the Perl script into the spreadsheet. Then you enter only two types of parameters:

- Signal frequencies for all signals.
- Output loading capacitance.

Output capacitive loading can be specified for each individual pin or specified as a default value for all pins.

Displayed in the spreadsheet (Figure 1) are three end results based on these two input parameters:

- Total I_{dd} consumed by the design.
- Total power consumed by the design.
- Total power consumed by output capacitance loading.

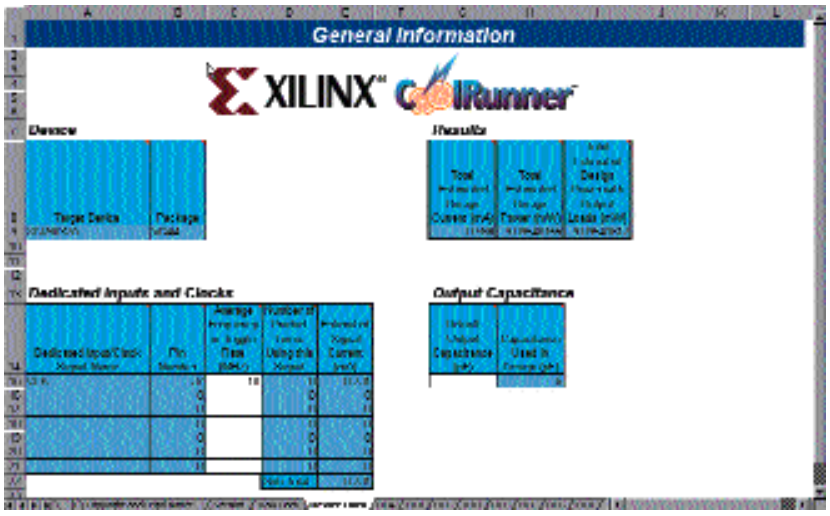


Figure 1 - Device data sheet showing results.

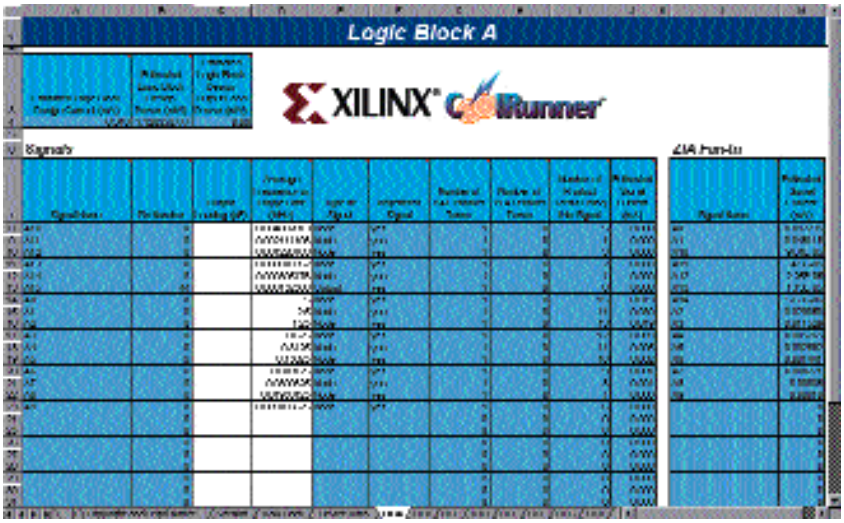


Figure 2 - Logic Block A sheet showing signal details.

There are estimates throughout the tool that show I_{dd} for specific architectural areas in the device as shown in Figure 2. For example, if you are attempting to reduce the overall device current, you can see what fast module or logic block is consuming the most current. Then it is easy to see what specific signals are demanding the most current within a logic block, because the current estimations for individual signals are shown.

In addition to the power estimation, this tool can assist you during the fitting process by displaying other important information. For example, statistical data for each individual signal is available including pin number, fan-in number of product terms, and fan-out number of product terms that load the signal. Fan-in data from the Zero Power Interconnect Array (ZIA) can easily be seen for each logic block as well as the fast module Global ZIA fan-in and fan-out data for XPLA2 devices.

This extra data is quite useful in determining the layout of a design within the CoolRunner device; you can effortlessly see the details of how the fitter arranged the design. Using this information, your design can be rearranged to better improve the timing performance.

Conclusion

The CoolRunner Power Estimator tool is an easy way to estimate power and I_{dd} information. Not only can you determine the power loading, you can also use this tool to help improve timing performance by visually interpreting the results in the spreadsheet.

Implementing an I²C Bus Controller in a *CoolRunner* CPLD

Here's an overview of a complete design that you can download from the Web.

by Jennifer Jenkins, Applications Engineer, Xilinx,
jennifer.jenkins@xilinx.com

The I²C bus is a popular serial, two-wire interface used in many systems because of its low electrical overhead. The two-wire interface minimizes interconnections so ICs have fewer pins and the number of traces required on printed circuit boards is reduced. The bus is capable of up to 100KHz operation, and each device connected to the bus is software addressable by a unique address with a simple master/slave protocol.

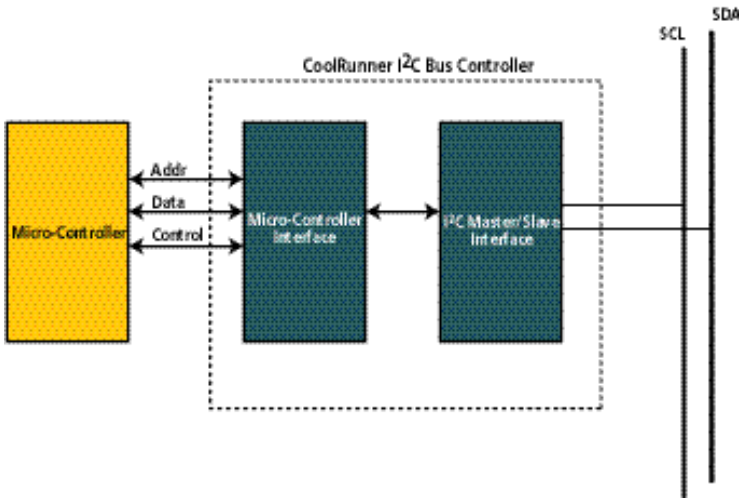


Figure 1 - CoolRunner I²C Bus Controller.

Designing with an I²C bus for handheld devices requires that you minimize power dissipation. That's why CoolRunner CPLDs, the lowest power CPLDs available, are the perfect devices for creating I²C controllers. The I²C design shown in this article provides both master/slave capability with an asynchronous byte-wide microcontroller or microprocessor interface as shown in Figure 1.

Functionality

The CoolRunner CPLD implementation of the I²C controller supports the following features:

- Microcontroller interface.
- Master or Slave operation.
- Multi-master operation.
- Software selectable acknowledge bit.
- Arbitration of lost interrupts with automatic mode switching from Master to Slave
- Calling address identification interrupt with automatic mode switching from Master to Slave.
- START and STOP signal generation/detection.
- Repeated START signal generation.
- Acknowledge bit generation/detection.

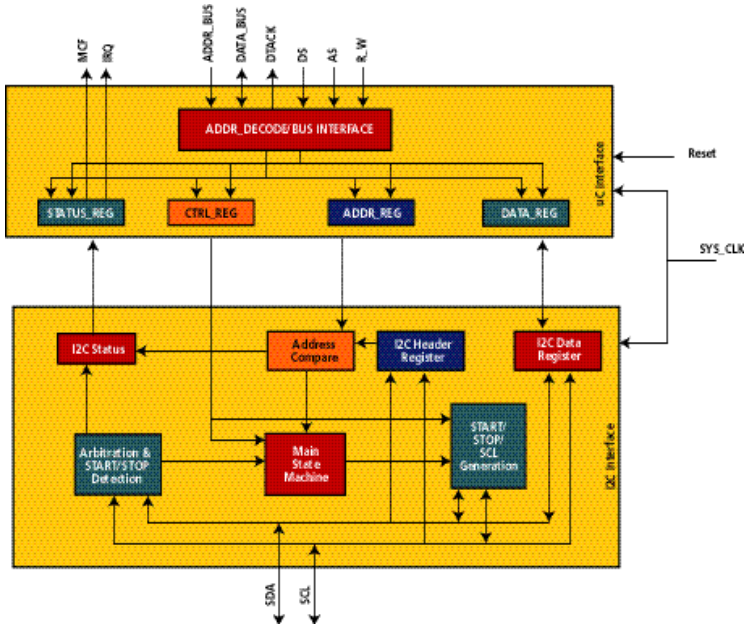


Figure 2 - CoolRunner I²C Controller block diagram.

- Bus busy detection.
- 100KHz operation.

Block Diagram

The functionality of the CoolRunner I²C controller is divided into two major blocks, the microcontroller interface and the I²C interface, as shown in Figure 2. This design was created in VHDL and verified through simulation. Xilinx software tools were used for compilation and fitting. The design was targeted to a 3V, 128-macrocell, enhanced clocking, CoolRunner CPLD in a 100-pin TQFP package (XCR3128A-10VQ100C).

Conclusion

This design offers a way to use an I²C bus in a low power application. Using a CoolRunner CPLD to implement the functionality of an I²C controller is perfect for power limited applications. The design of the I²C controller that is currently available for customers includes the following:

- Complete detailed application note
- Complete VHDL source code
- VHDL test benches

More information can be found at www.xilinx.com/apps/epld.htm,under CoolRunner XAPP315 - Implementing an I²C Bus Controller in a CoolRunner™ CPLD. The VHDL code and test benches created for this design are available by contacting Xilinx Technical Support at 1-800-255-7778.

How to Create an Portable Player Using a CoolRunner CPLD

CoolRunner CPLDs are used as the main controller for an MP3 player. This design is available for download from the Web.

by Anita Schreiber, CPLD Applications Engineer
Xilinx, anita.schreiber@xilinx.com

Portable MP3 players are the latest trend in music-listening technology; they require no moving parts, and can be built with very few components. And, because these devices are battery operated, they require low power technologies. That's why the new CoolRunner CPLD family from Xilinx is ideal for this application; this family offers advanced 3V and 5V devices with extremely low power dissipation.

Block Diagram

The block diagram of the CoolRunner MP3 Portable Player is shown in Figure 1. The shaded area shows the logic that is contained in the CoolRunner CPLD. All other blocks are external devices that can be obtained commercially.

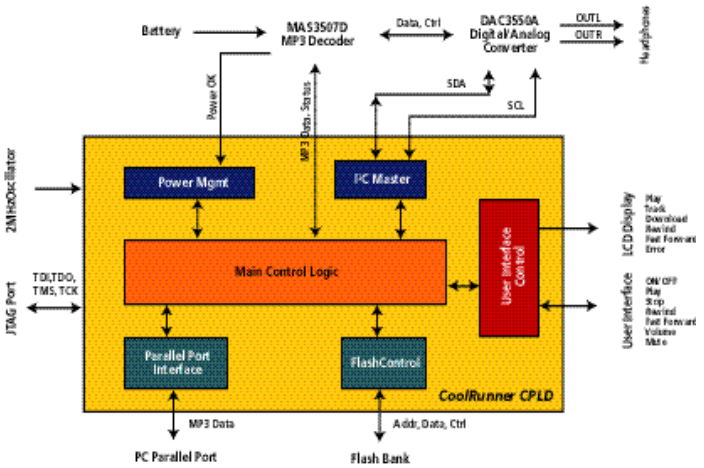


Figure 1 - MP3 portable player block diagram.

CoolRunner CPLD Logic Modules

The CoolRunner CPLD implements the following functions:

- **The Main Control Logic** provides the intelligence of the CPLD logic and controls all of the various functions.
- **The Parallel Port Interface** is used to downloading MP3 data files to the MP3 portable player.
- **The Flash Control** logic directs the Song Flash memory during MP3 data storage and retrieval. It also controls the Starting Address Flash memory which stores the beginning address of each song in the Song Flash memory.
- **The I^2C Master** logic block controls the DAC3550A Digital-to-Analog converter.
- **The User Interface Control** logic receives the user input and updates the LCD display.
- **The Power Management** logic monitors the power level reported

User Interface Button	Function
Play	Pressing this button allows the user to continuously play MP3 songs. When the last song has been played,the MP3 player begins playing the first song again.
Rewind	Pressing this button allows the user to skip to the beginning of the previous song.
Fast Forward	Pressing this button allows the user to skip to the beginning of the next song.
Stop	Pressing this button will halt playing of MP3 songs and resets the player to the beginning of the current song.
Volume/Mute	The volume can be increased and decreased as desired. The player can also be muted.

Table 1 - User interface functions.

from the internal DC/DC converter in the MAS3507D and insures the MP3 Portable Player shuts down properly when the battery voltage drops or the user wishes to turn the player off.

External Devices

The MAS3507D MP3 Decoder chip and the DAC3550A Stereo Audio DAC are both available from Micronas Intermetall and provide a complementary chip set for MPEG decoding and playback. The MAS3507D contains an embedded DC/DC converter which is used to supply the power to the entire player.

The Flash memory bank is composed of 32Mbytes of Flash memory for the storage of MP3 data files and 2Mbytes of Flash memory for the storage of the starting addresses of each song in the Song Flash memory. This design assumes that an LCD display could be designed with unique icons for the various MP3 user interface functions.

User Interface Functions and Operations

The design of this MP3 portable player assumes that a software package is designed that allows users to “rip” CDs and download a collection of their favorite MP3 files to this portable player. The user operations for listening to MP3 songs are shown in Table 1.

Conclusion

CoolRunner CPLDs provide the ideal programmable logic solution for any battery-operated or portable design. This design shows just one example of how a CoolRunner CPLD can be used in these types of applications to provide you with the many benefits of programmable logic while meeting the stringent power requirements of this market.

For more information,the following can be obtained from the Xilinx@Work section of the Xilinx website (http://xilinx.com/products/xaw/index.htm):

- Detailed Application Note (XAPP328)
- VHDL Source Code
- VHDL Testbenches

Get the Best Registered I/O Timing with Virtex-E FPGAs

You can achieve I/O setup times of less than 1.6 ns, and I/O clock-to-out times of less than 3.3 ns, using LVTTL switching levels.

by Randy Robinson, Xilinx, randy.robinson@xilinx.com

Virtex-E FPGAs contain a number of architectural features that enhance I/O timing. Four dedicated clock buffer networks allow low skew distribution of clocks to a large number of loads, while guaranteeing zero hold time requirements between registers. You can also use on-chip clock Delay Lock Loops (DLLs) to effectively eliminate the phase difference between a clock external to the FPGA and the internally buffered equivalent. Plus, each Virtex-E I/O block (IOB) contains both an input an output register, which can be used to improve I/O timing.

The setup time for a signal brought on chip is the sum of the I/O pad delay, any routing and logic delays, and the intrinsic setup time of the register or latch, minus any clock delay. Clock delay is the sum of I/O pad delay, any clock buffer delay, plus routing and loading delays. Of these items, only the I/O pad delays and register setup times are known precisely. All other factors are design dependent. Clock-to-out times require similar calculations, except that any clock delay worsens the clock-to-out time.

Registering Input Signals

Based on your system timing requirements you can allocate a setup time, hold time, and clock-to-out timing budget for the Virtex-E device. Typically, you are looking for the minimum required setup time for the FPGA, and zero (or negative) hold time.

The system-level timing specification references all input signal and clock timing values to the pins of the FPGA device. For example, a 5ns setup time budget means:

- Valid data will exist at the FPGA pin a minimum of 5ns prior to an active clock edge at the pin of the FPGA.
- The timing at the FPGA register used to capture this input signal must meet its own setup and hold time requirements.

It is mandatory that both of these requirements be met within the overall system timing budget.

You can alter input timing when using Virtex-E chips by controlling the following choices:

Selection of IOB and CLB

In Virtex-E FPGAs, input signal registers can be located in either IOBs or in CLBs. An advantage of using an IOB register is that the data path delay (consisting of I/O pad and buffer delays, plus routing) is fixed. If a CLB register is used, the routing delay will vary depending upon where the CLB is placed, and the routing path taken.

Using the appropriate timespecs can improve the setup timing when using CLB registers, but will not cause the Alliance Series™ or Foundation Series 2.1i software to move registered signals between IOB or CLB registers. If the source netlist contains register components with the property IOB = TRUE, these will not be moved. However, by either setting a MAP command line switch (map -pr i) or through enabling a GUI-based option to allow MAP to pack registers into the IOB cells, then a netlist which did not contain IOB registers can use IOB registers if possible.

Using DLLs

If a clock is distributed using a global buffer only (not a clock DLL) there is a delay between the clock at the FPGA input pin and the clock internal to the FPGA. This delay is beneficial in that it reduces the required setup time for the input signal. If a clock DLL is used, the propagation delay is effectively zero, and the equivalent system setup time is increased. This apparent disadvantage of using a clock DLL is offset by removal of the delay element (described in the next paragraph), and the improvement in clock-to-out times.

Using or Eliminating the IOB Delay Element

By default, a delay element is placed in the data path of the IOB. This delay block is used to guarantee a zero hold time for the IOB register, but has the effect of increasing the required setup time. If a clock DLL output is not used as the I/O register clock, and the delay element is removed, the setup time required will decrease dramatically, but now a positive hold time exists.

If you use a clock DLL, the delay element can safely be removed, giving you the dual benefit of less required setup time and a negative hold time. Beginning with the Alliance Series and Foundation Series 2.1i Service Pack Number 2, the delay element is included only when a DLL is not used and an IOB input register is used. This default behavior was changed from earlier software versions.

Registering Output Signals

When you are driving a registered signal off chip, you must meet a system clock-to-output specification. Fortunately, you have a number of simple options that can be used to meet this timing when using Virtex-E devices. The choices include:

Using IOB or CLB registers

You can choose whether the output register is an IOB register or a CLB register. The IOB register will always have a shorter routing path from register output to FPGA pin, so the fastest clock-to-out time will always be achieved by using the IOB register. Timespecs alone will not force the Alliance Series or Foundation Series 2.1i software to make the register type decision. However, by either setting a MAP command line switch (map -pr o) or through enabling a GUI based option to allow MAP to pack registers into the IOB cells, IOB output registers will be used, if possible.

	Run #1 Baseline	Run #2 Timespecs	Run #3 Add Pack IOB Registers	Run #4 Add NODELAY	Run #5 Add Fast Slew	Run #6 Add24 ma Drive	Run #7 Pick Best Results
Tsu (no DLL)	0.906 ns	0.442 ns	1.795 ns	-0.403 ns	-0.403 ns	-0.403 ns	1.795 ns
Tc2o (no DLL)	8.934 ns	9.082 ns	6.554 ns	6.554 ns	4.654 ns	4.453 ns	4.453 ns
Tsu (DLL)	2.407 ns	2.214 ns	1.555 ns	1.555 ns	1.555 ns	1.555 ns	1.555 ns
Tc2o (DLL)	7.837 ns	7.924 ns	5.325 ns	5.325 ns	3.425 ns	3.224 ns	3.224 ns

* The gray shading indicates a positive hold time requirement.

Table 1 - Runtime results.

Using Clock DLLs

Any delay of the input clock due to clock buffer and clock routing delays adversely effect clock-to-out timing, because this delay simply adds to the overall datapath delay. The use of a clock DLL will always result in faster clock-to-out times in a Virtex-E FPGA.

Using Slew Rate and Drive Strength Options for LVTTL Outputs

If LVTTL output buffers are selected, you can choose between seven drive strengths, and between fast and slow slew rate. By default, a Virtex-E output buffer defaults to LVTTL, 12ma drive, and slow slew rate. To improve clock-to-out times, select a higher drive strength (16ma or 24ma) and fast slew rate. The place and route software will not modify output buffer settings to meet a timespec; you must do this explicitly, typically by using a constraint file.

Summary of Recommendations

- Use IOB input and output registers when possible.
- Use clock DLLs when possible.
- Set fast slew rate and 24ma drive for LVTTL I/O to get fastest clock-to-out times.
- Set NODELAY mode when using clock DLLs to get shortest setup time.
- If clock DLLs are not used, NODELAY will greatly shorten your setup time, at the cost of a positive hold time.

Xilinx created a simple design to demonstrate the effects discussed above. The results are shown below.

Test Results

In Table 1 Run #1 shows the registered I/O timings using default settings and CLB registers. By adding timespecs (5ns setup, 10ns clock-to-out), results changed slightly (Run #2). Packing registers into IOBs (Run #3) gives a better clock-to-out delay, and a slightly worse setup time. The timing effects of setting NODELAY (Run #4), plus fast slew rate (Run #5), plus high drive (Run #6) are shown. Finally Run #7 gives the best results with zero hold time in all cases.

Conclusion

With new Virtex-E family you can easily achieve very fast registered I/O timing.

For complete information on Virtex-E FPGAs, see www.xilinx.com.

Understanding

Setup and Hold Times One Key to Successful Designs

Using synchronous design techniques is one essential key to creating reliable designs.

by Claude Gaschet, Xilinx Field Applications Engineer
Reptronic (France), claude@xilinx.com

The Xilinx software technology provides all the information you need to create reliable, high performance designs, but you should make sure that you don't violate two of the most important parameters: setup and hold time.

The Basics

Synchronous elements such as D flip-flops (DFF's) accept the data present on their D inputs when the clock transitions. However, the data must be stable prior to the clock edge (setup time, T_{su}) and maintained after the same clock edge (hold time, T_h); T_h can safely be ignored with Xilinx FPGAs (T_h is 0). See Figure 1.

If the setup and hold times are not violated, the data on the D input is transferred to the Q output, after the flip-flop clock-to-output delay (T_{cko}). However, if T_{su} or T_h timing is not met, the Q output is indeterminate.

Controlling Logic Paths

A real design is usually made of thousands of flip-flops, with several levels of logic between them, and moderate to high utilization of routing

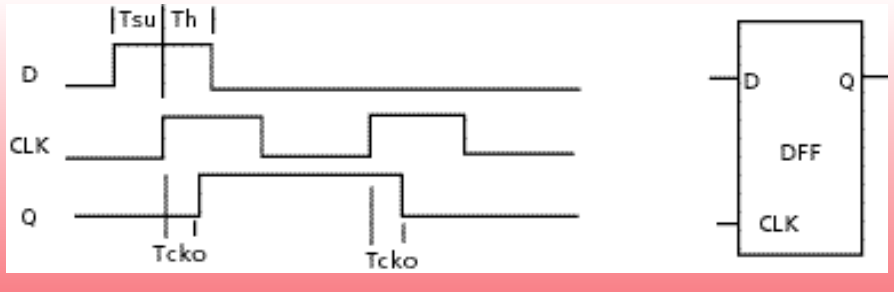


Figure 1 - Setup and hold timing.

resources. All those paths, from a data source element (PAD, FF, RAM...) to a receiving element (PAD, FF, RAM...), are known as logic paths.

Controlling those paths (giving the implementation tools the adequate directives to meet the timing criteria) is one key to success. In the Xilinx software tools, there are several graphical tools (Constraints Editor, FloorPlanner) to help you pass the right control information to the Implementation Engine. Another set of verification tools (the GUI or command-line-driven Timing Analyzer or TRCE) give you everything you need to check your results. But, what if some of your timing criteria are not met? Is your design failing because the timing constraint is not possible to meet ? If so why? Is it failing because of a setup time violation or a hold time violation? And, how do you fix the problem?

If your timing is OK, you'll get a Timing Analyzer message (in the .TWR report) that says "All Timing Constraints met." If your timing is not OK, you'll see messages that tell you why (such as "...clock frequency too high," or "...too many logic levels for the requested frequency," or " ...too much routing delays versus logic delays,") and you will be given the actual timing difference.

Two Examples of Clock Distribution

Consider a logic path starting at the Q output of a flip-flop, going through a number of logic stages, and ending at the D input of a another flip-flop, as shown in Figure 2. The safe operation is specified by the equation:

$$T_{clk} \geq T_{cko} + (\text{logic and routing delays}) + T_{su}$$

The Best Case - Minimal Clock Skew

In an ideal design, there will be minimal clock skew (all flip-flops see the clock edge at the same time, within a few hundred picoseconds). For synchronous designs, the optimum clock distribution is obtained when you use one of the dedicated clock routing resources attached to a global buffer (BUFG). Depending upon the FPGA you select, you have from two to eight BUFGs available.

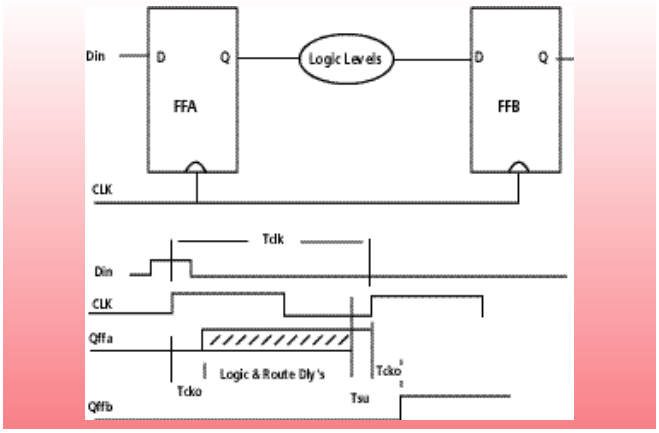


Figure 2 - Basic timing model.

For a given clock frequency, and for a given FPGA, you only have to control the number of logical levels (logic delays) the signal passes through, and make sure that this sum of delays does not exceed the clock period.

How to Correct the Problem

Figure 3 shows an excerpt of a report, showing a timing error. The report shows that we missed our timing goal (the slack value is negative, -0.662ns). We have a 67% logic budget for a 33% route budget (Xilinx recommends the opposite. A 50/50 ratio is generally OK for small design, and the bigger the chip, the bigger the recommended route budget (40/60 or 30/70).

The best solution is to decrease the number of logic levels; this will remove one combinatorial logic delay (T_{ilo}) and one net delay. An alternative would be to simply use a faster speed grade, because the you only need to gain 0.66ns.

```
Timing constraint:TS_CLK = PERIOD TIMEGRP "CLK"5 nS HIGH 50.000 % ;
1 item analyzed,1 timing error detected.
 Minimum period is 5.662ns.
--------------------------------------------------------------------------------

Slack:-0.662ns path DLY1 to $Net00012_ relative to
5.000ns delay constraint


Path DLY1 to $Net00012_ contains 4 levels of logic:
Path starting from Comp:CLB_R5C4.K (from $Net00004_)
To Delay type Delay(ns) Physical Resource
Logical Resource(s)
-------------------------------------------------------------------------------

CLB_R5C4.XQ Tcko 1.192R DLY1
 $I3/$1I37
CLB_R5C4.F1 net (fanout=1) 0.530R DIN_Q
CLB_R5C4.X Tilo 0.959R DLY1
 $I8
CLB_R5C5.F1 net (fanout=1) 0.832R DLY1
CLB_R5C5.X Tilo 0.959R DLY2
 $I9
CLB_R6C5.F1 net (fanout=1) 0.522R DLY2
CLB_R6C5.K Tick 0.668R $Net00012_
 $I10
 $I7/$1I37
-------------------------------------------------------------------------------

Total (3.778ns logic,1.884ns route) 5.662ns (to $Net00004_)
 (66.7% logic,33.3% route)
```

Figure 3 - A .TWR report showing a timing violation.

The Worst Case - Significant Clock Skew

In designs where the number of clocks exceeds the dedicated BUFG clock resources, some of the clocks will need to use the non-dedicated, standard, routing resources which will introduce clock skew. It's best to assign the fastest, high fanout, clocks to the dedicated BUFG routing resources. How does clock skew affect the design? And, how do you minimize its negative effects?

In Figure 4, the FFB clock is delayed from the FFA clock (skewed) and there is a direct connection from the Q output of FFA to the D input of FFB.

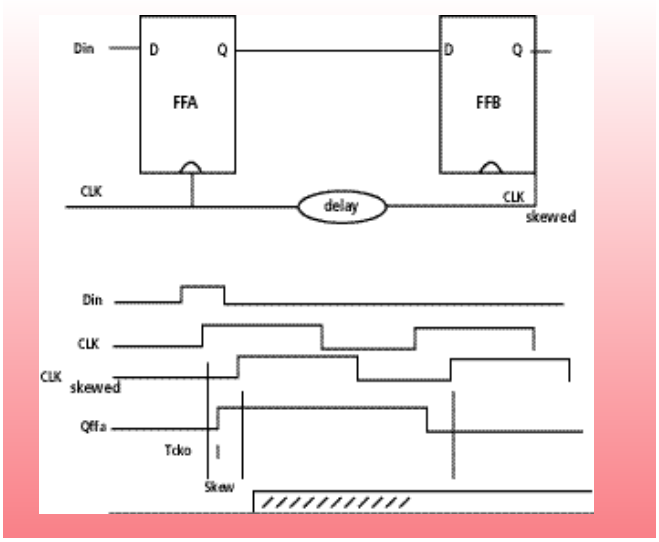


Figure 4 - Synchronous circuit with clock skew.

For FFB to clock in the correct data, you must respect the following condition :

$$T_{cko}(FFA) \geq SKEW$$

However, due to the clock delay, the D output of FFA may be unstable, changing, when FFB receives the clock edge, causing FFB to latch the wrong data. The CLK frequency does not affect the problem, therefore reducing the CLK speed won't solve the problem.

Once again, the .TWR report will give you the detailed information you need to identify and correct the problem. In the report, shown in Figure 5, you can quickly see that you missed your timing goal (the report shows negative slack, -3.234ns). The summary gives a (logic + route delay) of 3.06ns for a 6.6ns clock skew; The message "6.592ns skew between D_186 and D_188" should alert you to a probable data hold violation.

```
Slack: -3.234ns path D_186 to D_188 relative to
 6.592ns skew between D_186 and D_188

Path D_186 to D_188 contains 2 levels of logic:
Path starting from Comp: CLB_R45C25.S0.CLK (from clk_0)
To Delay type Delay(ns) Physical Resource
 Logical Resource(s)
-------------------------------------------------
CLB_R45C25.S0.YQ Tcko 1.065F D_186
 genck_0__genreg_185__ff
CLB_R45C10.S0.BY net (fanout=1) 2.127F D_186
CLB_R45C10.S0.CLK Tckdi -0.166F D_188
 genck_0__genreg_186__ff
-------------------------------------------------
Total (0.899ns logic, 2.127ns route) 3.026ns (to clk_0)
 (29.7% logic, 70.3% route)
```

Figure 5 - Timing report showing clock skew.

How to Correct the Problem

The easiest solution is to compensate for the clock skew by adding dummy logic in the data path to delay the data.

Another solution is to propagate clock and data in the opposite direction, as shown in Figure 6, (this may require manual edits, using the FPGA_Editor). You'll still get skew, and it will be reported in the .TWR, but you won't experience data loss.

Compared to the first equation, now you have the following relationship:

$$T_{clk}\text{-skew} = T_{cko} + (\text{logic and routing delays}) + T_{su}$$

This simply means that you now have less time for logic and routing delays, and you'll have to be careful not to violate the receiving flip-flop setup time, especially for high frequency operation.

Of course, the best solution would be to use BUFG to distribute the clock, which will eliminate the skew entirely.

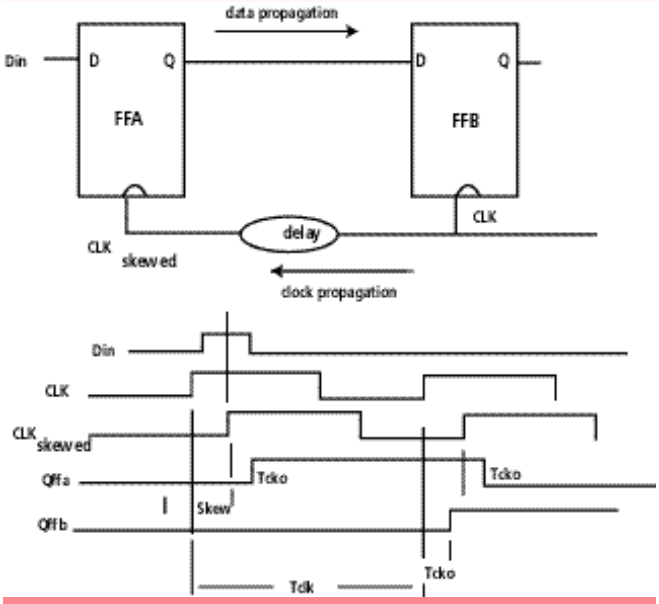


Figure 6 - Controlling the effects of clock skew through propagating the clock in the opposite direction.

Conclusion

The techniques discussed here apply to any type of logic design, not just programmable logic. However, it's easy to get reliable results with Xilinx components because they include special clock distribution networks (BUFGs). In our newer families (Virtex, Virtex-E, SPARTAN-II) we also include Delay Locked Loops that eliminate skew for both on-chip and off-chip clocks. Σ

If you need more help, go to www.support.xilinx.com

HDL Coding for PSEUDO-RANDOM Noise Generators

Inferring Virtex SRL macros results in extremely efficient Linear Feedback Shift Register implementations.

by Mike Gulotta, Field Application Engineer, Xilinx,
mike.gulotta@xilinx.com

Linear Feedback Shift Registers (LFSRs) are a fundamental function in applications such as pseudo-random noise (PN) generators, stream encryption, and error detection/correction. You can achieve extremely efficient LFSR implementations by using the Virtex Shift Register LUT (SRL). And, with today's Virtex-friendly synthesis tools, HDL code can be used to infer the SRL, thereby maintaining code portability.

PN Generators

PN generators are at the heart of every spread spectrum system, and are a good example for demonstrating how you can dramatically reduce FPGA utilization by exploiting the Virtex SRL. In a CDMA system, many PN generators are needed to distinguish channels, base stations, and handsets. Rake receivers, used in CDMA systems, consist of many copies of the same receiver, each called a finger, and each finger requires two PN generators, one for the "I" (In-phase) and one for the "Q" (Quadrature) channel.

Finding a way to improve the FPGA implementation efficiency of a circuit that is copied many times in a system (such as a PN generator), will obviously provide a huge savings.

Linear Feedback Shift Registers

Though the mathematics behind a PN code can be extremely complicated, the LFSR implementation can be relatively simple. A typical LFSR consists of a chain of registers and a modulo-2 adder (XOR gate). Predefined registers are "tapped" and fed to the XOR gate, and the XOR output is fed back to the first register in the chain, as shown in Figure 1. In a CDMA system, the predefined register taps are carefully determined to provided good auto correlation and cross correlation, and are often expressed as a polynomial such as, $P(x) = x^{17} + x^4 + 1$. An LFSR with n registers can sequence through $(2^n - 1)$ states. (See Xilinx XAPP 210 and XAPP 211 for additional information regarding LFSRs and SRLs).

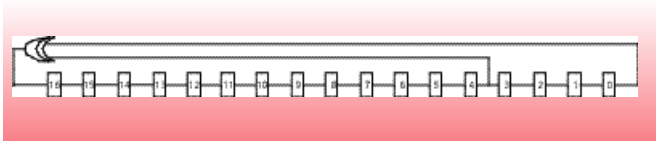


Figure 1 - A typical LFSR.

CDMA system requirements may require additional control of the basic LFSR. "Augmenting" the sequence, by adding an additional state, may be required to achieve 2^n states (instead of 2^n - 1), to maintain an even modulo count. Also, "puncturing" the sequence by periodically skipping a state may be required, if only a subset of the total 2^n states (such as 3 out of 4) are needed.

LFSR HDL Coding

The Virtex FPGA architecture is highly efficient for creating LFSRs. For example, the following code will infer a 64-bit shift register using Virtex SRLs rather than flip-flops (FFs).

VHDL	Verilog
process(clk)	Always @(posedge clk) begin
begin	Y <= {Y[62:0],INPUT};
if clk'event and clk='1' then	end
Y <= Y(62 downto 0) & INPUT;	
end if;	
end process;	

Figure 2 - HDL Code.

Using SRLs instead of FFs, this circuit will cost only one Configurable Logic Block (CLB) instead of 16. With such dramatic savings it is worth looking into ways to use SRLs whenever possible.

However, SRL registers cannot be loaded or read simultaneously, nor can they be asynchronously reset. In a PN generator application it may be necessary to jump out of sequence, which can be done by various techniques such as parallel loading the LFSR with a predetermined state. This can still be satisfied with the SRL by serially filling the LFSR with a predetermined state. To do this, a multiplexer is required in the LFSR feedback path allowing the loop to be broken while the predetermined state is shifted in, as shown in figure 3.

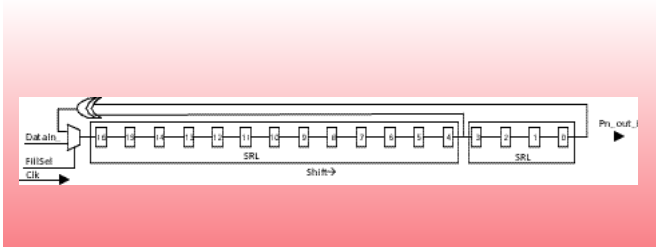


Figure 3 - PN generator.

The following verilog code implements an LFSR with several input controls that may be used to accommodate a PN generator application. The ShiftEn signal may be used to stall (augment) and/or puncture the sequence, and the FillSel and DataIn signals may be used to jump out of sequence. The `define compiler directive along with the Reset signal provide code portability by allowing the code to infer SRLs if targeting Virtex FPGAs, or to infer typical asynchronous reset FFs if targeting another technology. The number of taps are fixed, however the tap points and LFSR length are parameterized.

Conclusion

The Virtex architecture is very efficient for creating PN generators by using the Virtex Shift Register LUT (SRL). The SRL can also be used in many other applications such as pipeline balancers, filters, dividers, and waveform generators. In large systems, such as CDMA, the overall FPGA utilization can be reduced considerably by

```
/*************************************************************

The following is example code that implements an LFSR that can be used as
part of a pn generator.

The number of taps are fixed,however the tap points are parameratized.  The
LFSR length is also parameritized.  This code is not intended to be technology
specific.  When targeting Xilinx (Virtex) however, all the latest synthesis tools
(Leonardo, Synplify, and FPGA Express) will infer the Shift Register LUTS
(SRL16) resulting in a very efficient implementation.

Control signals have been provided to allow external circuitry to control such
things as filling,puncturing,stalling (augmentation),etc.  Only minimal simula-
tions have been run on this code as it is intended to be used for reference pur-
poses only.

A compiler directive can be used to steer the following code to infer typical FF's
(w/async resets),or, infer Virtex SLR16E elements.  Controlling the compiler
flow can be done by uncommenting the following line.  This can also be done
from a top level module.

*************************************************************/
// `define non_Virtex_device  // Comment out to infer Virtex SRL16s.
module pn_gen_lg_srl (clk,pn_out_i,ShiftEn,FillSel,DataIn_i,RESET);
parameter Width = 17;  // LFSR length (ie. number of storage elements)
// Parameratize l channel LFSR taps.
// l(x) = X**17 + X**4 + 1
parameter l_tap4 = 4;  // l channel LFSR,single tap.
//Ports
input   clk,DataIn_i,FillSel,ShiftEn,RESET;
output  pn_out_i;

// l channel ////////////////////
reg [l_tap4-1:0] srl1_i;
reg [Width-1:l_tap4] srl2_i;
wire  lfsr_in_i,par_out_i;

assign   pn_out_i = srl1_i[0];
assign   par_out_i = srl2_i[l_tap4] ^ srl1_i[0];
assign   lfsr_in_i = FillSel ? DataIn_i :par_out_i;

`ifdef non_Virtex_device // compiler directive, if defined,will infer asyn reset FF
always @(posedge clk or negedge RESET) begin
   if (!RESET) begin
      srl1_i <= 0;
      srl2_i <= 0;
   end
   else
`else  // compiler directive, if not defined,will infer SRL16.

always @(posedge clk)  begin

`endif

   if (ShiftEn) begin
      srl2_i <= {lfsr_in_i,srl2_i[Width-1:l_tap4+1]};
      srl1_i <= {srl2_i[l_tap4],srl1_i[l_tap4-1:1]};
   end
end

endmodule
```

When targeting Virtex,all the latest synthesis tools (Leonardo, Synplify, and FPGA Express) will infer the SRL16E resulting in a very efficient implementation. The compiler directive, used to steer the code to infer FFs or Virtex SLR16E elements, may not be supported by all synthesis tools. Controlling the compiler flow can be done by un-commenting the first line. (This can also be done from a top level module). Only minimal simulations have been run on this code as it is intended to be used for reference purposes only.

taking advantage of the SRL, which can lead to smaller, fewer, and less expensive parts. With only a basic understanding of the SRL, along with today's Virtex-friendly synthesis tools, these savings can be accomplished easily and without sacrificing code portability.

Post Synthesis Verification for Virtex FPGAs

For large designs especially, verification throughout the design flow will save you time and effort.

Nij Dorairaj, Sr. CAD Engineer, Exemplar Logic, Inc., nij@exemplar.com

As FPGAs grow bigger and faster it becomes increasingly important to verify your designs; this saves you time and produces better designs. Verification after synthesis should be part of the design process because this ensures that you pass the correct netlist to the Xilinx place and route tools. Also, you can debug your designs faster, whether you are using an RTL netlist (RTL verification) or a synthesized netlist (post synthesis verification).

Post Synthesis Verification

A synthesis tool usually writes a netlist based on the UNISIM library. The cells in the library are modeled well for both simulation and synthesis. For example in the verilog UNISIM library, the LUTs (look up tables) are modeled using UDP (user defined primitives). This can speed up your design simulation considerably, and because most of the combinational logic will be mapped to LUTs, overall simulation will also be fast. In short, if the synthesis tool can write a Verilog/VHDL netlist with LUT cells and their INIT attributes, verification will be very accurate and fast.

Enhancements in Leonardo Spectrum

Traditionally, verification was performed on the Verilog or VHDL netlist generated by the Xilinx place and route tools. Thanks to new functionality in the latest release of LeonardoSpectrum from Exemplar Logic, gate-level verification is now supported prior to place and route, using the Xilinx UNISIM simulation library.

Procedure for Using Gate-level Simulation

A special variable in LeonardoSpectrum must be set, to turn on this feature:

```
xi_write_init_on_luts (default is FALSE)
```

This variable must to be set to “TRUE” before writing out the Verilog/VHDL netlist. The variable can either be set in the GUI, using Tools -> Variable Editor, or in the interactive shell you can type:

```
set xi_write_init_on_luts TRUE
```

Verilog Example

Here is a simple Verilog example, to demonstrate the feature. The RTL design is synthesized in LeonardoSpectrum and a Verilog netlist (with the above variable set to TRUE) is written out. The synthesized netlist is simulated using the MTI (Modeltech) Verilog simulation tool. The design output is the registered value of A and B and C. The register clock is four times slower than the main clock; it uses CLKDLL to divide the main clock by four. The output waveform is shown in Figure 1.

Notice the INIT properties being used to configure the LUT cells of the synthesized netlist.

RTL Design

```
// module defintion of CLKDLL
module CLKDLL ( CLKIN, CLKFB, RST, CLK0,CLK90,CLK180,CLK270,
         CLK2X,CLKDV, LOCKED
         ) ;
 parameter CLKDV_DIVIDE = "2.0" ;
 parameter DUTY_CYCLE_CORRECTION = "TRUE";
 input  CLKIN, CLKFB, RST ;
 output      CLK0,CLK90,CLK180,CLK270,CLK2X,CLKDV, LOCKED ;
endmodule // CLKDLL

module slow_clk_dll (CLKOUT, CLKIN, RST);
 input CLKIN, RST;
 output CLKOUT;
 wire IBUFG_OUT, BUFG_A_IN, BUFG_A_OUT, BUFG_B_IN, BUFG_B_OUT;
 IBUFG Ub_ibufg ( .I(CLKIN),.O(IBUFG_OUT) );
 CLKDLL  Ub_clkdll  (  .CLKIN(IBUFG_OUT),.CLKFB(BUFG_A_OUT),.RST(RST),
           .CLK0(BUFG_A_IN),.CLKDV(BUFG_B_IN)  );
 // divide the input clock by 4
 defparam Ub_clkdll.CLKDV_DIVIDE = "4.0";
 BUFG Ub_bufgA ( .I (BUFG_A_IN),.O(BUFG_A_OUT) );
 BUFG Ub_bufgB ( .I (BUFG_B_IN),.O(BUFG_B_OUT) );
 assign CLKOUT = BUFG_B_OUT;
endmodule // slow_clk_dll

module simple (A,B , C, RST, OUT, CLK);
 input A, B, C, RST, CLK;
 output OUT;
 reg  OUT;
 wire SLOW_STEADY_CLK;
 slow_clk_dll  simple_clk  (SLOW_STEADY_CLK,CLK,RST);
 always @ (posedge SLOW_STEADY_CLK)
  begin
    if (RST)
     OUT = 1'b0;
    else
     OUT = A & B & C;
  end
endmodule // simple
```

Synthesized Netlist

```
module simple ( A, B, C, RST, OUT, CLK ) ;
  input A , B, C, RST, CLK;
  output OUT ;
  wire  SLOW_STEADY_CLK,simple_clk_BUFG_B_IN, simple_clk_BUFG_A_OUT,
    simple_clk_BUFG_A_IN, simple_clk_IBUFG_OUT, A_int,B_int,C_int,
    RST_int,OUT_dup0,nx53,nx54;
  wire [4:0] \$dummy ;
  IBUFG simple_clk_Ub_ibufg (.O (simple_clk_IBUFG_OUT),.I (CLK)) ;
  CLKDLL simple_clk_Ub_clkdll (.CLK0 (simple_clk_BUFG_A_IN),.CLK90 (
     \$dummy [0]),.CLK180 (\$dummy [1]),.CLK270 (\$dummy [2]),.CLK2X (
     \$dummy [3]),.CLKDV (simple_clk_BUFG_B_IN),.LOCKED (\$dummy [4]),.CLKIN (
     simple_clk_IBUFG_OUT),.CLKFB  (simple_clk_BUFG_A_OUT),.RST  (RST_int)
     ) ;
     defparam simple_clk_Ub_clkdll.CLKDV_DIVIDE = 4.0;
     defparam simple_clk_Ub_clkdll.DUTY_CYCLE_CORRECTION = "TRUE";
  BUFG simple_clk_Ub_bufgA (.O (simple_clk_BUFG_A_OUT),.I (
    simple_clk_BUFG_A_IN)) ;
  BUFG simple_clk_Ub_bufgB (.O (SLOW_STEADY_CLK),.I (simple_clk_BUFG_B_IN)) ;
  OBUF OUT_obuf (.O (OUT),.I (OUT_dup0)) ;
  IBUF RST_ibuf (.O (RST_int),.I (RST)) ;
  IBUF C_ibuf (.O (C_int),.I (C)) ;
  IBUF B_ibuf (.O (B_int),.I (B)) ;
  IBUF A_ibuf (.O (A_int),.I (A)) ;
  FDR reg_OUT (.Q (OUT_dup0),.D (nx54),.C (SLOW_STEADY_CLK),.R (nx53)) ;
  LUT2 ix46 (.O (nx53),.I0 (C_int),.I1 (RST_int)) ;
    defparam ix46.INIT = 4'hD;
  LUT2 ix47 (.O (nx54),.I0 (A_int),.I1 (B_int)) ;
    defparam ix47.INIT = 4'h8;
endmodule
```

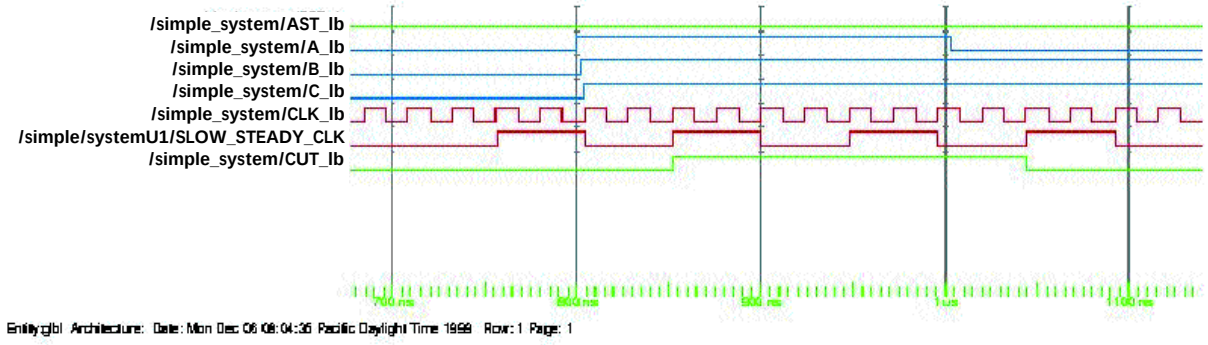


Figure 1 - Output waveform from Modeltech.

Conclusion

With LeonardoSpectrum you can verify the correctness of your netlist before you go to place and route, and get high quality results for your Virtex FPGA designs.

A more complete version of this article, with a test bench,is available at: http://www.exemplar.com/support/appnotes.html.

References

1. Virtex chapter in LeonardoSpectrum technology manual (leo_tech.pdf). 2. Verilog GSR/GTS Simulation Methodology in Xcell issue 33 (3rd quarter 1999).

ModelSim™

Using the ModelSim FPGA Library Manager

Using the new FPGA Library Manager will improve your simulation time by easily building Xilinx FGPA libraries for use within ModelSim.

by Joe Rodriguez, Technical Marketing Engineer, Model Technology Inc., joer@model.com

ModelSim is a mixed-language, single-kernel simulator, allowing you to simulate Xilinx FPGA instances implemented in Verilog, together with other instances in VHDL, in the same simulation run. You have flexibility in your choice of HDL language, and you can easily build the simulation library for your chosen Xilinx FPGA (and HDL language) with the FGPA Library Manager which supports simprim, unisim, and LogiBLOX simulation libraries in both Verilog and VHDL for all Xilinx FPGA families.

Using the FPGA Library Manager

The new ModelSim FPGA library manager guides you through the process of compiling the FPGA libraries provided by Xilinx. These compiled libraries can then be used with your design to simulate with ModelSim. The source code for the Xilinx libraries is included with the Alliance Series Software, and is referenced by the "XILINX" Environment Variable.

Here are step by step instructions for using the Library Manager:

- Invoke ModelSim and go to a working directory where you want to compile the FPGA libraries (this can be where your design source files are located). A welcome banner will appear. This new banner contains several new features, including a new Project Wizard. The Project Wizard is documented in the ModelSim Reference Manual, and in the Tutorial which has an example project. Dismiss the Welcome Banner, continue to ModelSim.
- From the main ModelSim window change directories. Use **File > Change Directory** to Browse for your working directory
- From the main window command line enter: **vlib work**
- Select the FPGA Library Manager which will build the FPGA library. Use **Design > FPGA Library** Manager.
- Use the Browser to select the Xilinx TCL script. (ModelSim uses TCL as a scripting language.) The Library manager uses a support file that resided in the ModelSim_install_directory/contrib. These scripts will be updated as the Xilinx parts list evolves. See the Additional Information section below for updated script downloads. Use **Browse > Select fpgavendor_xilinx.tcl > Open > Next** (Please refer to Figure 1 "ModelSim XILINX FPGA Library Manager Window".)
- Select a ModelSim project initialization file

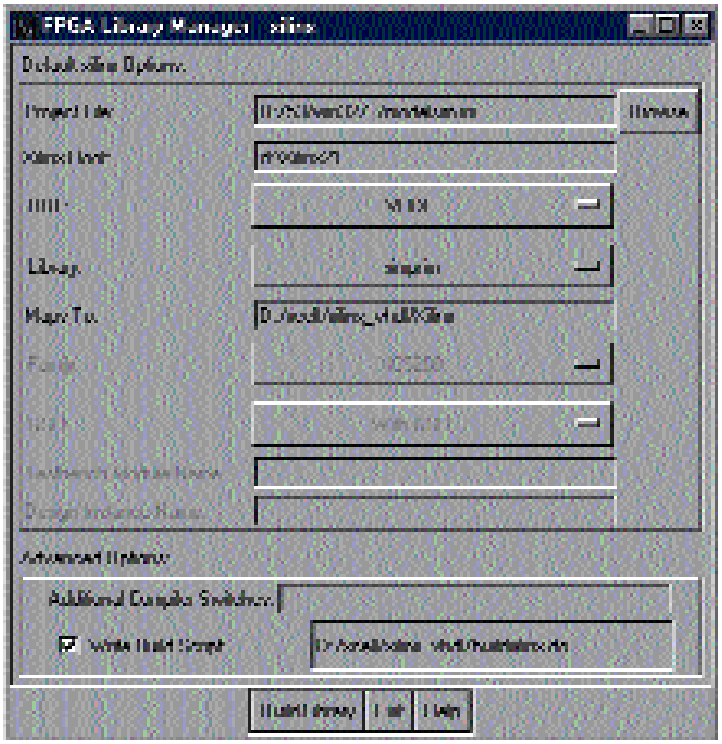


Figure 1: ModelSim Xilinx FPGA library manager window.

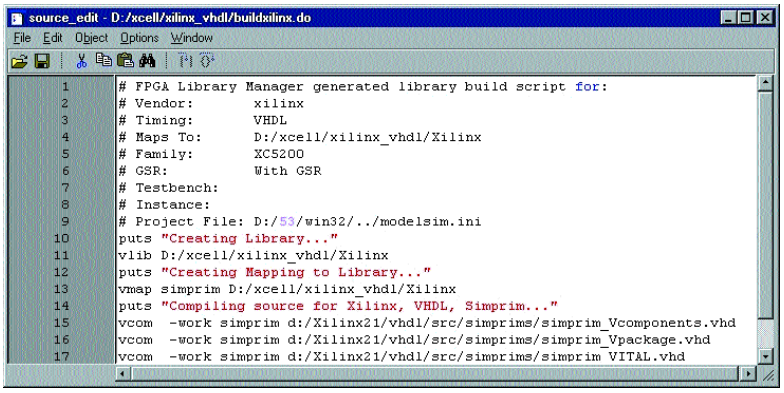


Figure 2: ModelSim Xilinx FPGA library script.

(which contains default settings for ModelSim). The Xilinx root directory value is from your system environment variables.

- Choose your HDL language (VHDL or Verilog). If you have a VHDL testbench, you can run a Verilog description of your design with ModelSim, if the pin names match exactly (d[0:31] does not match d1,d2....d31) and you are using STD LOGIC at the IO (ModelSim will map VHDL to Verilog strength).
- Select the Xilinx FPGA Library. The Xilinx Library source code that gets compiled is from the Xilinx root directory, which is read from your system environment variable. The example uses the XC5200 Family. The Maps To entry is a name of the ModelSim library you wish to use. This can be any name you wish, default is XILINX. This physical name can be mapped to the needed logical name required by the source code.
- Select **Write Build Script**; the default path name is the current directory. (ModelSim is a full featured simulation tool that can be run in UI mode, command line mode, or batch mode. The ModelSim FPGA Library Manager can create a script that can be used to re-compile the FPGA library.) The TCL script can be seen in Figure 2: the vlib command creates a ModelSim library, vcom is the VHDL compiler, and

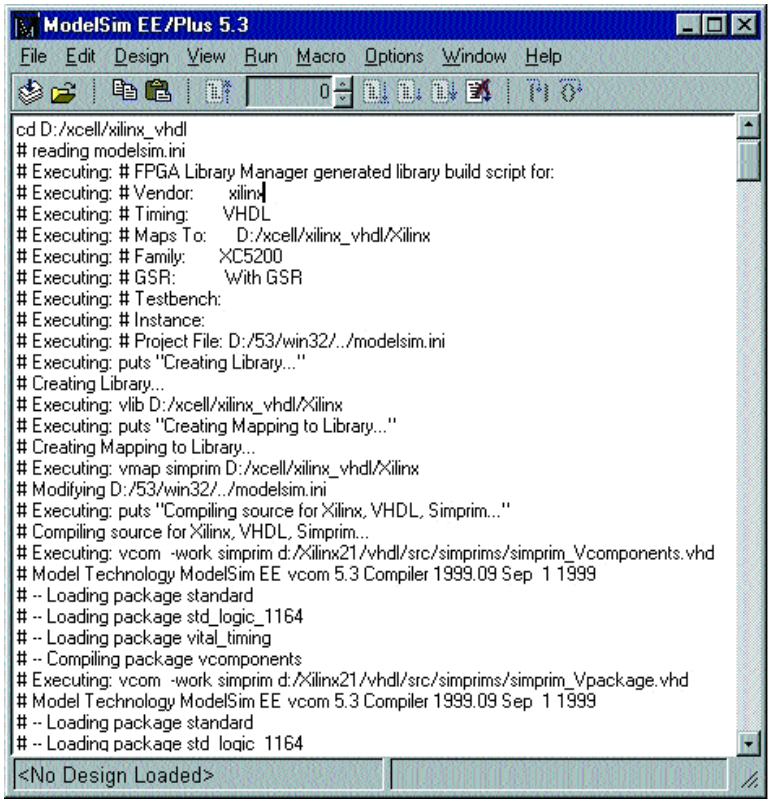


Figure 3: Main ModelSim window.

vmap maps the physical location to the logical name used in your VHDL source code. This script can now be used to build the library without the use of the FPGA Library Manager. See the MTI Application Notes for a full description of the use of scripts with ModelSim.

- Select **Build Library**. The Main Window scrolls messages indicating the library has been built as shown in Figure 3. When the library compilation is complete an information window will appear. Dismiss the library compiled information window, and Exit the FPGA Library Manager. The ModelSim build script is shown in Figure 2.

Main Window -> Open ->Open Source, select buildxilinx.do.

The library is now available for use with your source files.

Additional Information

For updates to the Xilinx FPGA vendor compile scripts, go to:

http://www.model.com/resources/fpgalibmgr.html

For a complete Application Note ModelSim 5.2 With Xilinx Alliance 2.1 showing step by step instructions for using ModelSim to compile and simulate any of your Xilinx designs, go to:

http://www.model.com/pdf/113_xilinx_21.pdf

For a list of all MTI Application Notes, including ModelSim 5.2 with Alliance 1.5, and example circuits for use with Application Notes, go to:

http://www.model.com/support/technote/index.html

Conclusion

The new ModelSim FPGA library manager removes the question of how to compile the Xilinx FPGA libraries, so you can focus on the verification of your Xilinx design.

FPGA System

Simulation and Synthesis

Using *Synopsys VCS and FPGA Compiler II*

This HDL design methodology can help you use the largest Virtex FPGAs with a minimum amount of time spent on synthesis, simulation, and verification.

by Srikanth Vijayaraghavan, Applications Consultant, Synopsys,
raghavan@synopsys.com

Synopsys has combined its powerful logic synthesis technology with innovative architecture-specific optimization technology to address the needs of FPGA designers who are now adopting an HDL methodology. FPGA Compiler II delivers powerful architecture-specific synthesis capabilities with features such as behavioral re-timing and pipelining capability. Synopsys VCS (Verilog Compiled Simulator) provides a fully-featured implementation of the verilog language as defined in the IEEE Standard Hardware Description Language (IEEE Std 1364-1995).

VCS is specifically designed to simulate large, complex designs faster than any other Verilog-HDL simulator. VCS supports interfaces to a variety of other simulators and models, including (but not limited to) user PLI applications conforming to IEEE Std 1363-1995, delay calculators, SDF delay annotation, LMG Smatmodels, and the LMSI hardware modeler.

The combination of the FPGA Compiler II synthesis tool and the VCS simulator provides a simple and accurate design and verification flow that significantly reduces your total development time.

Detailed Design Flow

The steps involved in taking a design from an RTL description to a production FPGA are summarized in Figure 1. These steps are explained in detail using the RTL description of a flip-flop shown in Example 1. After synthesizing the RTL code for this flip-flop using FPGA Compiler II, you can automatically generate a functional Verilog simulation netlist as shown in Example 2. FPGA Compiler II is capable of synthesizing the design, either by flattening the design completely, or by preserving the complete hierarchy.

```
Module reg1 (clk,reset,din,dout);
input clk,reset;
input  din;
output dout;

wire clk,clk_enbl,reset;
wire  din;
reg  dout;

always@(negedge clk or posedge reset)
if (reset)
                        dout = 0;
else
                        dout = din;
endmodule
```

Example 1 - RTL description of a simple flip-flop.

```
// Synopsys FPGA Compiler II
// automatically generated file
// Author: raghavan
// Program:FPGA Compiler II
// Version:3.2.0.4206

module FDC_1 ( Q ,D ,C ,CLR );
    output Q ;
    input D ;
    input C ;
    input CLR ;
    wire synch_enable ;
    reg Q ;
    always@( negedge C or posedge CLR )
    begin
        if ( CLR ) Q = 1'b0;
        else Q = ( D );
    end
assign synch_enable = ( 1'b1 );
endmodule

module IBUF ( O ,I );
    output O ;
    input I ;
    assign O = ( I );
endmodule

module OBUF_S_12 ( O ,I );
    output O ;
    input I ;
    assign O = ( I );
endmodule

module reg1 ( clk ,reset ,din ,dout );
    input clk ;
    input reset ;
    input din ;
    output dout ;

    wire N_clk ;
    wire N_reset ;
    wire N_din ;
    wire N_dout ;

FDC_1 dout_reg (.CLR (N_reset),.Q (N_dout),.C (N_clk),.D (N_din));
IBUF C_clk (.I (clk),.O (N_clk));
IBUF C_reset (.I reset),.O(N_reset ) );
IBUF C_din (.I (din),.O (N_din));
OBUF_S_12
C_dout (.I (N_dout),.O (dout));

Endmodule
```

Example 2 - Post synthesis functional netlist generated by FPGA Compiler II.

The netlist can be simulated in VCS like any other Verilog file. FPGA Compiler II maintains the port names at the module level and therefore debugging through the hierarchy is very easy.

Assuming you have a testbench for the design module named test_reg1.v, a simple command line script for simulation in VCS will look as follows:

```
vcs -RI -Mupdate reg1.v test_reg1.v -o reg1.simv -l reg1.log
```

Where:

- **-RI** is for simulating and bringing up the XVCS Debugging debugging GUI automatically.
- **-Mupdate** is to enable incremental compile.
- **-o** is to provide a distinct name for the executable file; default name is simv.
- **-l** is to provide a distinct name for the log file produced during compile.

Once the functionality of the design is verified, the EDIF equivalent to this Verilog netlist is generated using FPGA Compiler II. The timing constraints entered in the FPGA Compiler II GUI can be exported into a spec file (.ncf), which is understood by the Xilinx software tools. The Xilinx software uses this EDIF netlist and the .ncf constraint file to place and route the design.

If the design routes successfully, you can write a Verilog simulation netlist in terms of gate cells, and also a delay file (.sdf, Standard Delay Format). A part of the delay file that corresponds to the gate level netlist is shown in Example 3. This .sdf file should be included during simulation to back annotate the original delays into the design.

The delay values need to be back annotated through the PLI interface in VCS. The .sdf file is called from the gate-level netlist through the "$sdf_annotate" utility. For performing a gate-

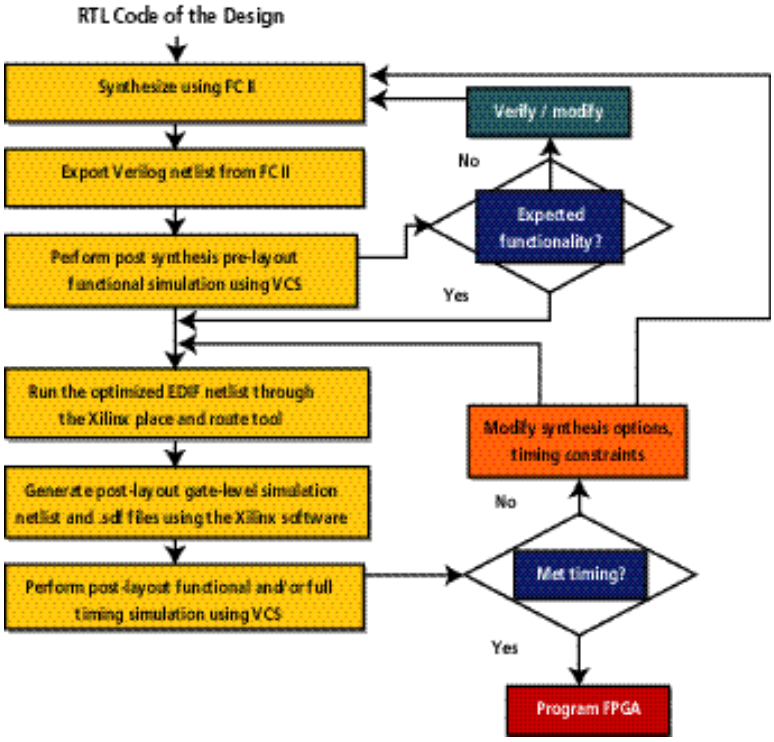


Figure 1 - Detailed design f.low for Xilinx FPGA's

```
(TIMESCALE 1 ps)
  (CELL
    (CELLTYPE "X_FF")
      (INSTANCE dout_reg)
        (DELAY
          (ABSOLUTE
            (PORT IN (1948:1948:1948) (1948:1948:1948))
            (PORT CLK (1520:1520:1520) (1520:1520:1520))
            (PORT RST (0:0:0) (0:0:0))
            (IOPATH CLK OUT (887:887:887) (887:887:887))
            (IOPATH SET OUT (887:887:887) (887:887:887))
            (IOPATH RST OUT (887:887:887) (887:887:887))
          )
        )
(TIMINGCHECK
(SETUP (negedge RST) (posedge CLK) (479:479:479))
(SETUP (posedge IN) (posedge CLK) (195:195:195))
(SETUP (negedge IN) (posedge CLK) (195:195:195))
(HOLD (posedge IN) (posedge CLK) (0:0:0))
(HOLD (negedge IN) (posedge CLK) (0:0:0))
(WIDTH (negedge CLK) (3456:3456:3456))
(WIDTH (posedge CLK) (3456:3456:3456))
(WIDTH (posedge RST) (3456:3456:3456))
        )
  )
```

Example 3 - SDF file generated by Xilinx software.

level simulation, you need to first create a table file. This table file will list all the PLI tasks that need to be included. The "simprims" directory inside the Xilinx installation contains the Verilog description of all the library cells (.vmd extension). These descriptions contain the timing checks for the setup and hold time violations. Annotating the original delay values from the .sdf file during simulation performs these checks.

Now create a pli.tab file, with the following content:

```
$sdf_annotate call=sdf_annotate_call
acc+=mp,prx:reg_gate+
```

Where: reg_gate is the name of the top level module

Now, to perform a gate level simulation, you can use a sample script as follows:

```
vcs -RI -Mupdate reg_gate.v test_reg.v
-o gate_reg -l gate_reg.log \
-y $XILINX/verilog/src/simprims \
+libext+.vmd+ \
-P pli.tab
```

Where:

- **-y** stands for the library directory.
- **+libext+.vmd+** stands for all the files with extension .vmd.
- **-P** stands for pli table file.

Conclusion

The Synopsys FPGA Compiler II and VCS provide an easy and intuitive HDL design methodology, a seamless design flow, and high level control within the design process. In addition, using FPGA Compiler II gives you the power to effectively use Xilinx Virtex devices with the highest quality results.

digital SET-TOP boxes

An Ideal Application

for SPARTAN FPGAs

Digital set top boxes are a catalyst for consumer-based computing, and they hasten the shift towards a networked home.

by Rebecca Burr, Manager of Market Analysis,
Xilinx, burr@xilinx.com

According to market researchers, the digital set-top box (DSTB) market is projected to grow at a compound annual growth rate approaching 17% over the next few years. Production is anticipated to double between 1999 and 2002, to over forty million units a year.

Today, most set top boxes deliver minimal functionality, and the market remains fragmented with varying levels of features and disparate standards. In addition, cable suppliers are seeking to expand services and features, and include the functions currently offered by other consumer appliances, to justify increased fees. This is an ideal application for the flexibility, low cost, and time-to-market benefits of Spartan FPGAs.

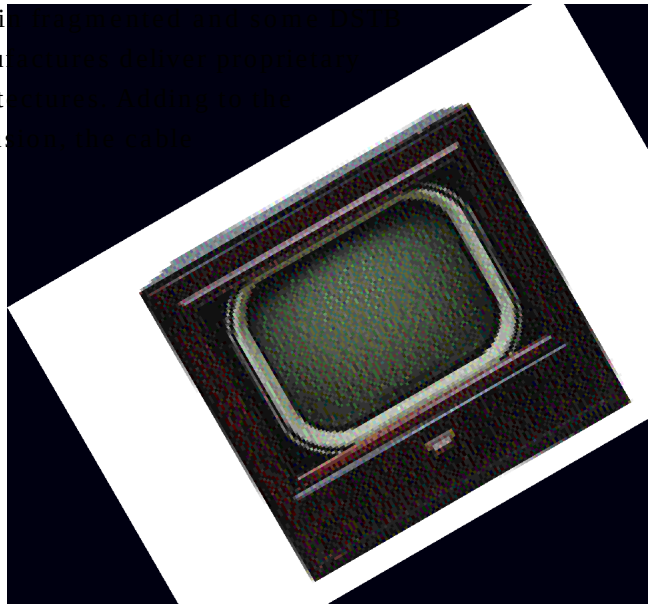
Possible Features

Some of the possible features that can be offered by DSTBs:

- Phone caller ID displayed to the television screen.
- E-commerce for grocery shopping, Web browsing, and banking.
- Infotainment such as interactive games, Video on Demand, on-line casinos.
- E-mail.
- Videoconferencing.
- Downloading music files from the internet.
- Recording television programs.

Market Structure

In the set top box market, a small number of producers dominate. However, standards still remai
manu
archit
confus



market is essentially a subscriber-based business where the service providers furnish the hardware for free. Recently, the digital satellite industry also began adopting this approach by offering a rental option for set-top boxes.

The Collective DSTB Market

The DSTB market is composed of three competing technologies:

- **Digital Terrestrial** systems use conventional antennas, and are already in limited use in Europe, with broader service launches expected over the next few years. Japan also expects to launch Digital Terrestrial services in the 2003 timeframe. Because it leverages the existing infrastructure, this technology is expected to exhibit explosive growth.
- **Digital Cable** is a conversion of existing analog cable distribution systems, and broadcasters have been swift to implement these standards.
- **Digital Satellite** technology is still in its infancy, offering lower cost distribution, service delivery to thinly populated regions, and ease of upgradeability. Standards for Digital Satellite systems are not consistent between broadcasters.

Conclusion

DSTB designers must optimize for both price and performance in an increasingly competitive marketplace. Plus, they must provide more functionality and create robust designs that offer flexibility for varying standards and for backward and forward compatibility. This is an impossible task for ASICs, and that's why the Spartan FPGA family is a key component of any DSTB strategy.

Questions & Answers

From the Xilinx Applications Engineering Staff

by Rohit Sawhney, Product Applications Manager, Xilinx, rohit@xilinx.com

Virtex I/Os

How do I pull the I/O pins to 5 Volts using external pullups?

To pull up the I/O pins to 5V externally, only three I/O standards are allowed: LVTTL, LVCMOS, and PCI33_5.

The internal I/O pullup can be considered to be a 50K resistor to Vcco (for 3.3-V or 2.5-V devices). If you try to pull an I/O to a voltage higher than Vcco, you need to do so with a resistor, with a smaller value than 50K. The devices have circuitry to disable the internal pullup if the I/O is pulled higher than about Vcco + 0.7V (a threshold voltage higher than Vcco), so if you pull an I/O up to 5V, there will be no static current into Vcco. But if you don't pull up the I/O past Vcco + 0.7V, then the internal pullup remains enabled and will "fight" with the external pullup.

Our recommendation is to use a 4.7K pullup resistor, but any value less than 4.7K will also work.

Software Installation

What do I do if my Alliance Series or Foundation Series 2.1i installation does not complete as a result of one of the following issues:

- The splash screen appears, then disappears without an error message.
- An unexpected error occurs during setup.

There may be several reasons for either of the error messages to appear. In general, ensure that you have administrative privileges and that there is plenty of hard drive space on the destination drive. Additionally, a re-boot is sometimes needed to release any locked Windows DLLs.

If you have done the above, and the installation still fails, the most likely problem has to do with the way the registry settings for certain environment variables are interacting with the 2.1i installer. Typically, simply resetting these variables will over-write the registry entry that is causing this problem, and allow the installer to succeed.

The variables that have been seen to exhibit the problem are: PATH, TEMP, and TMP.

- **Win9x**: Make sure these variables are listed in your autoexec.bat file.
- **WinNT**: You can check your environment by executing:
  'Start -> Settings -> Control Panel -> System -> Environment Tab'

In the System Variable section, there should be a PATH variable listed. In the User Variable section, there should be both TEMP and TMP variables. Re-updating each of these variables will correct the setting in the registry.

If there is no PATH variable listed in the User section, then create one. To do this:

1. Click on the Variable line and type in PATH.
2. Click on the value, and type in %PATH%.

By doing the above, the registry entries should get corrected, allowing the installation to succeed. For further information please reference:

- http://support.xilinx.com/techdocs/7074.htm
- http://support.xilinx.com/techdocs/7362.htm

CORE Generator

Are there any cores to speed up the implementation of Virtex designs?

The following cores are either included in the Xilinx 2.1i software release, or can be downloaded from the Xilinx CORE Generator Cores and IP Updates page: (http://www.xilinx.com/ipcenter/coregen/updates.htm#updatesCurrent)

The following cores are included in the 2.1i Release CD:

- Dynamic Constant Coefficient Multiplier.
- Divider.
- Block Memory modules (single and dual port).

The following cores can be downloaded from the Xilinx CORE Generator Cores and IP Updates page:

- Gate functions (AND, NAND, OR, NOR, XOR, XNOR, INVERTER).
- Multiplexer functions (bit, bus, slice BUFE/BUFT).
- Register functions (flip-flop and latch based).
- Parallel Multiplier (pipelined and combinatorial).
- Sine/Cosine Lookup TAble.
- Distributed Memories (single and dual port RAM and ROM).
- Adder/Subtractor.
- Accumulator.
- Comparator.
- Binary Counter.
- Decoder.
- FD-Based Shift Register.
- RAM-based Shift Register.
- Two's Complement.
- Numerically Controlled Oscillator (Single and Dual Channel)
- Distributed Arithmetic FIR Filter.
- Asynchronous FIFO.
- FFT and Inverse FFT (16-, 64-, 256-, and 1024-point).

Note: All Xilinx CORE Generator LogiCORE modules that support the Virtex architecture also support Virtex-E and Spartan-II architectures.

Synthesis

How can I instantiate a CLKDLL using Exemplar Spectrum or Synplicity?

Currently, CLKDLL has to be instantiated or manually inserted in Exemplar and Synplicity.

- A sample design and procedure for CLKDLL insertion in Exemplar Spectrum is documented in the following solution: http://www.xilinx.com/techdocs/7737.htm
- A sample design for CLKDLL instantiation in Synplicity is documented in the following solution: http://www.xilinx.com/techdocs/8144.htm

How can I infer a BlockRAM in Exemplar or Synplicity?

Both Exemplar Spectrum and Synplicity now support Virtex BlockRAM (a fully synchronized RAM) inference.

- Xilinx Solution 7929 (http://www.xilinx.com/techdocs/7929.htm) provides VHDL/Verilog example for Exemplar Spectrum.
- Xilinx Solution 2508 (http://www.xilinx.com/techdocs/2508.htm) provides VHDL/Verilog example for Synplicity.

Note: The BlockRAM inference example provided in XCELL 32 for Leonardo Spectrum was incorrect.

How can I infer a ROM in Synplicity?

One of the new features in Synplify 5.3 is ROM inference. Earlier versions of the Synplify compilers were able to infer ROM tables. Synplify 5.3 now maps these inferred ROMs to Xilinx ROM primitives (ROM16X1 and ROM32X1) with the use of an attribute called syn_romstyle.

Xilinx Solution 8183 shows how to infer these ROMs for the 4K and Virtex families in VHDL/Verilog: http://www.xilinx.com/techdocs/8183.htm.

Simulation

How do I use glbl.v module in the Verilog simulation?

In the 2.1i Alliance Series, the general procedure for specifying global signals for the Verilog simulation flow involves defining the global signals with the **$XILINX/verilog/src/glbl.v** module. This module allows a global signal to be modeled as a wire in a global module.

The glbl.v module connects the global signals to the design, which is why it is necessary to compile this module with the other design files and load it along with the toplevel.v file or the testbench.v file for simulation.

Configuration & JTAG/ Boundary Scan

What are the supported Xilinx cable, software, and device combinations?

The following combinations are supported, as described in http://support.xilinx.com/techdocs/8097.htm:

1. MultiLINX cable with Hardware Debugger v2.1i (or later):

- Supports slave serial configuration of XC3000A, Spartan/XL, XC4000E/EX/XL/XLA/XV, and Virtex/E devices.
- SelectMAP configuration on Virtex/E devices.
- Readback verify supported for XC4000E/EX/XL/XLA/XV, XC9500/XL/XV, Spartan/XL, and Virtex/E devices.

2. MultiLINX with JTAG Programmer v2.1i sp3 (or later):

- Supports JTAG configuration of XC1804, XC4000E/EX/XL/XLA/XV, XC9500/XL/XV, Spartan/XL, and Virtex/E devices.
- Supports Readback-Verify of the XC9500/XL/XV devices.

3. Parallel Cable III with Hardware Debugger v2.1i (or later):

- Supports slave serial configuration of XC3000A, XC4000E/EX/XL/XLA/XV, Spartan/XL, and Virtex/E devices.

4. Parallel Cable III with JTAG Programmer v2.1i sp3 (or later):

- Supports JTAG configuration of XC1804, XC4000E/EX/XL/XLA/XV, XC9500/XL/XV, Spartan/XL, Virtex/E devices.
- Supports Readback-Verify of the XC9500/XL/XV devices.

5. XChecker with Hardware Debugger v2.1i (or later):

- Supports slave serial configuration of XC3000A, XC4000E/EX/XL/XLA/XV, Spartan/XL, and Virtex/E devices.
- Supports Readback-Verify and Readback-Capture of XC4000E/EX/XL/XLA/XV, and Spartan/XL devices with a configuration bit stream of 250,000 bits or less.

6. XChecker with JTAG Programmer v2.1i sp3 (or later):

- Supports JTAG configuration of XC1804, XC4000E/EX/XL/XLA/XV, XC9500/XL/XV, Spartan/XL, Virtex/E devices.
- Supports Readback-Verify of the XC9500/XL/XV devices.

Q: Where can I find information on performing JTAG configuration on a Virtex device?

See Application Note 139 (Xapp139) Configuration and Readback of Virtex FPGAs Using (JTAG) Boundary-Scan at: http://www.xilinx.com/xapp/xapp139.pdf.

Q: What cable should be used with the Coolrunner ISP programmer?

The Xilinx Parallel III Cable should be used. Please reference: http://support.xilinx.com/techdocs/7588.htm.

Q: Which CoolRunner devices support ISP or Boundary Scan Operations?

The Coolrunner ISP and Boundary Scan support is listed in the following reference: http://support.xilinx.com/techdocs/8173.htm

support.xilinx.com

Q: What recent improvements have been made to the support.xilinx.com search engine?

A number of improvements have been made to our Web search engine:

- The first improvement was to update the underlying search algorithm. Previously we were AND-ing search terms together, which returned very precise results but often made it difficult for users to find what they were looking for. We decided to implement a more Internet friendly algorithm by using an 'ACCRUE' operation. This allows user queries to be scored, returning documents matching the highest number of keywords at the top of the list. Documents containing only one or two keywords would be returned at the bottom.
- The second improvement was to collect our older answer records into a separate archive. When new Xilinx users search our database, they no longer have to worry about sorting through older answer records to find the most current information. However, you will still have access to this older information.
- Finally, we've included the ability to search our software manuals from the same interface. It is now possible to search the Answers Database and online software documentation at the same time. Σ

Xilinx Trade Show Programs

Attend these events and see for yourself Xilinx technology at work.

by Darby Mason-Merchant, Trade Show Manager, Xilinx, darby@xilinx.com

Xilinx started the year with two new product releases addressing low power requirements-Spartan-II FPGAs and CoolRunner XPLA3 CPLDs. At the Portable Design 2000 and Wireless Symposium 2000 events, Xilinx demonstrated how our technology is "Spanning the Low Power Spectrum" with new wireless connectivity and MP3 Player demonstrations.

You can also expect to see numerous new partnerships and solutions, demonstrated at more shows and events worldwide. You'll also see where Xilinx is making a difference in applications such as telecom, communications, consumer, automotive, medical and more. FPGAs are everywhere and so is Xilinx!

Year 2000 Worldwide Xilinx Trade Show Schedules

Year 2000 North American Trade Show Schedule

Jan 25-26	Portable Design 2000	San Diego, CA
Feb 22-24	Wireless Symposium 2000	San Jose, CA
Feb 9-11	FPGA Conference 2000	Monterey, CA
March 14	Synopsys User's Group 2000	San Jose, CA
March 21-22	IP2000	Santa Clara,CA
April 10-12	DSP Spring Conference 2000	San Jose, CA
April 2000	FCCM Conference 2000	Napa,CA
May-Dec	Embedded Computing Shows	US & Canada
May 23-24	AC Developers Conference 2000	Santa Clara,CA
June 5-7	37th Design Automation Conference	Los Angeles, CA
June 21-23	WITI Technology Summit 2000	Santa Clara,CA
July 24-28	NSREC 2000	Reno, NV
Sept 6-7	Embedded Internet Conference 2000	San Jose, CA
Sept 26-28	MAPLD 2000	Laurel,MD
Oct 17-18	NCF / InfoVision 2000	Chicago, IL
Oct 18-20	Frontiers in Education 2000	Kansas City, MI

Year 2000 Worldwide Xilinx Trade Show Schedules

Year 2000 European Trade Show Schedule

Nov 2000	Electronica 2000	Munich,Germany

Year 2000 South East Asian Trade Show Schedule

March 23-24	IIC 2000	Guanzhou,China
March 27-28	IIC 2000	Shanghai,China
March 20-21	IIC 2000	Beijing,China
May 3-4	IIC Taipei	Taipei, Taiwan
October 2000	EDA&T	Hsinchu,Taiwan
October 2000	EDA&T	Beijing, China

Year 2000 Japanese Trade Show Schedule

Jan 27-28	EDA Techno Fair 2000	Tokyo, Japan
Nov 2000	Micon System Tool Fair 2000	Tokyo, Japan

For more information about Xilinx Worldwide Trade Show Programs, please contact one of the following Xilinx team members or see our website at: http://www.xilinx.com/company/tradeshows.htm

- **US Shows:** Darby Mason-Merchant at: darby@xilinx.com or Evangeline Tanner at etanner@xilinx.com.
- **European Shows:** Andrea Fionda at: andrea.fionda@xilinx.com.
- **Japanese Shows:** Tetsuo Souyama at: tetsuo.souyama@xilinx.com
- **SouthEast Asian Shows:** Mary Leung at: mary.leung@xilinx.com

Continued on page 62.

Go to the Xilinx web site to get the latest product information:
http://www.xilinx.com/products/products.htm

Development System Solutions

Xilinx offers a variety of development system solutions, enabling the design and implementation of Xilinx Programmable Logic devices. These development systems combine the industry's fastest place and route technology, with a flexible and easy to use graphical interface to help you achieve the best possible designs within your project schedule, regardless of your experience level.

Xilinx Alliance Series Software

http://www.xilinx.com/products/alliance.htm

The Alliance Series development systems are designed for customers who have made an investment in a customized EDA environment. The Xilinx Alliance Series software integrates into these environments by leveraging open systems standards, interfaces, and formats such as EDIF, SDF, VHDL, VITAL/Verilog, and STAMP. Combining the strengths of our EDA partners' tools with our advanced implementation features gives you the ultimate in flexibility and design performance.

Xilinx Foundation Series Software

http://www.xilinx.com/products/found.htm

The Foundation Series development systems are designed for customer's who are interested in purchasing a complete, ready-to-use design solution for all of their programmable logic needs. Xilinx Foundation Series solutions enable both new and experienced programmable logic designers to achieve handcrafted results automatically, through push-button design flows. Foundation Series solutions support schematic-only, HDL-only, and mixed-level design flows by seamlessly integrating some of the industry's best EDA tools.

ModelSim Xilinx Edition

ModelSim™

http://www.xilinx.com/products/software/mxe.htm

The ModelSim Xilinx Edition (XE) simulator is a complete HDL simulation environment, optimized for use in verifying Xilinx programmable logic designs. ModelSim XE enables designers to verify the source code (VHDL and Verilog), as well as the functional and timing models of their design using a common "self-checking" testbench. ModelSim XE provides a powerful first step into the world of HDL simulation with capacity and performance designed for the verification of the Xilinx XC9500 CPLD and Spartan series FPGA devices as well as lower-density XC4000 and Virtex series FPGAs.

ModelSim XE is most valuable for customers who understand the benefits of VHDL or Verilog simulation, and are looking for a cost-effective solution for low-density programmable logic design. It is available in both VHDL and Verilog versions. ModelSim XE may only be used with Xilinx v2.1i development systems and later. Xilinx sells the ModelSim XE products as options to any of its development systems.

CPLD Web-powered Software Solutions

The Xilinx CPLD Web-powered Software Solutions offer designers the flexibility to do CPLD design evaluation and fitting on-line or on their desktop. The WebFITTER is an on-line device fitting and evaluation tool which accepts VHDL, Verilog, ABEL, or netlist files. The WebPACK downloadable desktop solutions offer free CPLD software modules from ABEL and HDL synthesis to device fitting and JTAG programming.

WebFITTER

http://www.xilinx.com/sxpresso/webfitter.htm

The Xilinx WebFITTER is a free, Web-based CPLD design evaluation and fitting software tool that allows system designers to target their designs using the industry's best CPLDs, the XC9500 Series and the CoolRunner Series, on the latest version of Xilinx software and get their results and pricing in minutes.

WebPACK

http://www.xilinx.com/sxpresso/webpack.htm

The Xilinx WebPACK contains free downloadable software solutions for Xilinx XC9500 and CoolRunner Series CPLDs. Each solution provides a simple and intuitive design environment for any Xilinx CPLD family. The WebPACK is a collection of three design suites: design entry, device fitting, and programming. These tools can be downloaded and used individually or, when installed together, become an integrated design environment for Xilinx CPLDs.

2100 Logic Drive
San Jose, CA 95124-3450

Q100

First Class Presort
U.S. Postage
PAID
Permit No. 2196
San Jose, CA